MASTER EN SCIENCES INFORMATIQUES

Optimisation d'une Commande d'Acier par Programmation Linéaire

Castiaux, Jean-Pierre

Award date:
1993

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix

Institut d'Informatique
Rue Grandgagnage, 21
B-5000 Namur

Optimisation d'une Commande d'Acier par Programmation Linéaire

Mémoire réalisé pour l'obtention du grade de
Licencié & Maître en Informatique
par
Castiaux Jean-Pierre

Promoteur : Leclercq Jean-Paul, F.U.N.D.P.
Co-promoteur : Lescrenier Marc, ARBED Luxembourg.

Année académique 1992-1993

OBJET :

Ce document traite de l'analyse et de la conception d'une application informatique permettant l'optimisation d'une commande d'acier pour une aciérie produisant des poutrelles de longueurs variables.

L'application informatique élaborée fait appel à la théorie mathématique de la programmation linéaire. Elle consiste principalement en un programme linéaire autour duquel s'intègrent différents modules informatiques destinés à gérer les données et les résultats du programme linéaire.

Le document précise tout d'abord en quoi consiste une commande d'acier. L'analyse complète du programme linéaire ainsi que des différents modules sous-jacents est ensuite détaillée. Après est traité le problème des données en nombre trop important, engendré par l'utilisation de la programmation linéaire.

Le document propose également une préétude de l'aspect temps réel de la commande d'acier.

ABSTRACT :

This paper deals with the analysis and conception of a computer application intended to optimise a steel order for a plant producing steel beams of different lengths.

The designed computer application is based on the mathematical linear programming theory. It mainly consists of a linear program into which different information modules are integrated wich allow the management of the linear program data and results.

The paper firstly describes the steel order. The complete analysis of the linear program as well as the sub-programs is then detailed. It then looks at the problem of numerous data management in the linear program.

An approach of real time aspects of this application is also dealt with.

Remerciements

Je tiens à remercier toute l'équipe de l'Unité Modélisations & Optimisations pour l'accueil chaleureux, la disponibilité et la sympathie témoignées à mon égard tout au long de mon stage. Plus particulièrement je remercie, Monsieur Raymond Bausch, responsable de l'Unité, à qui il a toujours été agréable de présenter l'état d'avancement du projet qui m'a été confié en collaboration avec Marc Lescrenier; Marc Lescrenier, responsable du projet, qui m'a guidé vers une analyse fructueuse, pour la confiance témoignée et ses précieux conseils. Merci aussi à toutes les personnes travaillant pour ARBED que j'ai eu le plaisir de rencontrer dans le cadre ou en dehors du projet.

Je remercie également Monsieur Jean-Paul Leclercq, promoteur de ce mémoire, pour m'avoir proposé ce stage, pour le temps qu'il a pu consacrer à la lecture de ce mémoire, pour sa critique judicieuse et ses conseils dans les différentes étapes de la rédaction.

Sommaire

I. Introduction

A. Contexte du stage.

1. ARBED dans le monde de la sidérurgie.

C'est au monde sidérurgique auquel appartient le groupe ARBED que se rapporte ce document. Composé de quelques 400 sociétés, le groupe ARBED représente le cinquième groupe sidérurgique en Europe et figure parmi les 15 premiers producteurs des pays à économie de marché. ARBED S.A. Luxembourg a été fondée en 1882. Elle est la société mère du groupe dont les activités s'étendent de l'extraction minière à la transformation de l'acier. Les sociétés du groupe ARBED S.A. Luxembourg emploient plus de 50000 personnes et produisent plus de 7 millions de tonnes d'acier brut par an, dont près de la moitié au Luxembourg où 10000 personnes sont employées. Les autres sites de production sont localisés en Belgique, en Allemagne et au Brésil. Cela représente de quoi construire 1000 tours Eiffel par an, c'est-à-dire trois tours Eiffel chaque jour. L'acier qui est transformé en produits finis dans les laminoirs est commercialisé par TradeARBED, la principale organisation de vente du groupe. Son réseau compte plus de cinquante points d'appuis situés sur les cinq continents.

Sur le territoire luxembourgeois, ARBED possède cinq usines et un centre de recherche. Chacune des usines est spécialisée dans une gamme distincte de produits. Ces usines sont établies à Esch-Belval, Esch-Schifflange, Differdange, Dudelange et Rodange.

2. Le service informatique à ARBED S.A. Luxembourg.

C'est au sein d'une équipe du service informatique d'ARBED S.A. Luxembourg que mon stage s'est déroulé. Le service informatique, placé sous la direction de Monsieur Thill, est divisé en quatre départements :

- Infocentre : aide aux non informaticiens
- Systèmes départementaux : gestion des ordinateurs décentralisés
- Exploitation centrale : gestion de l'ordinateur central et du réseau
- Développement : programmation d'application

En particulier, le département Développement comporte six unités de développement d'applications :

- Circuit Client
- Circuit Production

- Comptabilité & Finance
- Personnel
- Modélisations & Optimisations
- Executive Information Systems

C'est l'équipe de l'Unité Modélisations & Optimisations dirigée par Monsieur Raymond Bausch et comprenant 8 personnes que j'ai eu le plaisir de renforcer durant la période de mon stage, sous la direction de Monsieur Marc Lescrenier.

3. L'application de la Commande d'Acier.

Un des vastes projets sur lesquels travaillent les membres de l'Unité Modélisations & Optimisations est l'optimisation de la chaîne de production des usines du groupe établies au Luxembourg, depuis l'analyse des commandes reçues jusqu'à l'expédition des produits finis. La Commande d'Acier constitue une des applications faisant partie de ce vaste projet aux objectifs ambitieux.

Très précisément, l'objet du document concerne l'optimisation, par la voie de la programmation linéaire, de la commande d'acier de l'aciérie d'Esch-Belval. La commande d'acier est liée de près au processus de fabrication. Pour comprendre en quoi elle consiste précisément et ce que l'on attend de son optimisation, une large partie de ce document est consacrée à sa description au sein du projet global. L'application Commande d'Acier étant cernée, la suite du document sera dédiée à une description détaillée de son optimisation.

B. Regard sous l'angle de la mathématique.

L'optimisation de la commande d'acier consiste en fait en un problème d'optimisation mathématique du processus de fabrication de poutrelles d'acier par le biais d'un ordonnancement dans leur production. Cette optimisation a pour objectif la minimisation de la quantité d'acier nécessaire à la fabrication des poutrelles.

Le sujet traité se résume à un problème d'affectations multiples, avec contraintes, au sein d'un programme linéaire.

Afin de brièvement expliquer le problème posé, il est nécessaire de donner quelques mots d'explication sur le procédé de fabrication. Les produits finis sortant de l'aciérie sont des poutrelles. On peut supposer ici : que toutes les poutrelles fabriquées sont strictement identiques à leur longueur près. Ces poutrelles sont fabriquées à partir de la découpe de barres-mères, qui ne sont rien d'autre que d'immenses poutrelles. Ceci est suffisant pour comprendre le problème du point de vue mathématique. Etant donné un ensemble de poutrelles à produire, il faut déterminer le plan de découpe des barres-mères en poutrelles le plus avantageux permettant de produire toutes les poutrelles de l'ensemble donné. Le plus avantageux signifie ici qui vérifie toutes les contraintes imposées et tel que la longueur totale d'acier utilisée soit minimale. Vu autrement, ce problème revient à constituer des barres-mères à partir des poutrelles à produire, de la façon la plus avantageuse. Nous dirons qu'il faut affecter les poutrelles à des barres-mères. Une contrainte imposée est notamment que les barres-mères peuvent être de taille quelconque dans un large intervalle, mais qu'au plus 3 tailles sont acceptées. Plusieurs méthodes peuvent être envisagées pour résoudre ce problème. Au vu de la puissance logicielle et matérielle actuellement disponibles sur le marché, une méthode envisagée est la voie de la programmation linéaire. En effet, s'il est possible de modéliser le problème et toutes les contraintes sous-jacentes, un avantage certain de la méthode est l'optimalité de la solution. Cependant, le temps calcul nécessaire est susceptible d'être trop élevé. Chaque méthode présente bien sûr des avantages et des inconvénients. Pour cette raison, une équipe s'est penchée sur la résolution du problème par utilisation d'heuristiques. L'utilisation d'heuristiques a rapidement mis en évidence un gain potentiel appréciable. Par la suite et parallèlement à cette méthode, la voie de la programmation linéaire a été approfondie. Ce document rapporte l'analyse du problème par la voie de la programmation linéaire. Celle-ci s'est en effet rapidement avérée très efficace et a finalement été privilégiée dans ce contexte précis.

C. Composition du document.

L'introduction de ce document constitue le premier chapitre.

Le second chapitre est dédié à la description du contexte dans lequel s'insère le problème posé et à son introduction. Au sein de ce chapitre seront abordés respectivement les points suivants : la description du produit fini que sont les poutrelles d'acier, la nomenclature, la planification de production des commandes, la Commande d'Acier, l'optimisation de la Commande d'Acier.

Le problème posé étant clairement précisé, le chapitre trois concerne l'optimisation de la commande d'acier du point de vue mathématique. Quelques mots au sujet de la voie de la programmation linéaire précèdent la modélisation pas à pas du problème.

Le quatrième chapitre concerne la description des modules nécessaires et externes au P.L. Ces modules sont gérés par une application globale dont le P.L. constitue le noyau.

Vient ensuite le chapitre cinq. Il parle du traitement des grands couples, c.-à-d. du traitement du problème lorsque les données sont trop nombreuses pour être gérées au sein du Programme Linéaire.

Enfin, dans le chapitre six se trouve la spécification complète de l'application globale.

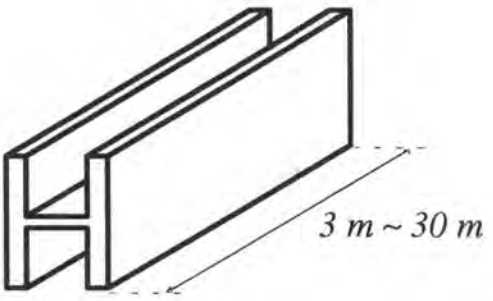
En annexe de ce document se trouve une préétude de l'aspect temps réel du problème traité.

II. Prélude à la Commande d'Acier

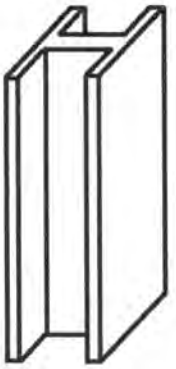
A. Une aciérie produisant des poutrelles.

Le produit fini :

Les produits finis de l'aciérie d'Esch-Belval sont des poutrelles d'acier aux dimensions variables et de qualités d'acier différentes:



Ainsi, les poutrelles produites peuvent présenter des aspects très différents en fonction de leur usage:



Pilier de fondation

Rail de chemin de fer

Ne nous laissons cependant pas surprendre par la diversité des aspects : ce ne sont toujours que des poutrelles, mais aux dimensions et qualité d'acier différentes. Le processus de fabrication reste identique. Pour l'instant, notre première réflexion concernant l'optimisation de la fabrication des poutrelles sera limitée à supposer que, pour des raisons de réglage des scies et des machines, il est préférable de fabriquer en série des poutrelles présentant des caractéristiques communes.

Le processus de fabrication :

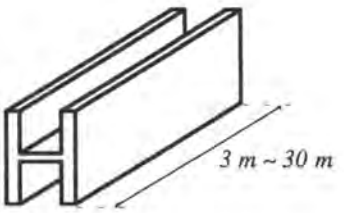
Penchons-nous sur le processus de fabrication des poutrelles. Cela nous permettra de mieux comprendre la modélisation de son optimisation.

Les *poutrelles* sont issues de la découpe transversale de barres-mères, qui ne sont rien d'autre que d'immenses poutrelles de l'ordre de 85 mètres. Une *barre-mère* est produite par laminages successifs d'un bloc d'acier appelé *lingot*. Les lingots eux proviennent d'immenses chaudrons, appelés *lingotières*, dans lesquels est coulé l'acier selon des quantités établies liées aux dimensions des barres-mères désirées. Une restriction porte cependant sur le procédé de laminage des lingots en barres-mères. Ce procédé est tel qu'une des dimensions caractérisant les barres-mères (et donc les poutrelles issues) sera fixée et restera invariable pour une durée de travail de huit heures. Cette dimension devant être fixée est la distance entre les ailes des barres-mères, c'est-à-dire si l'on caricature une barre-mère par un H, la distance séparant les pieds du H. Durant une session de travail de huit heures, toutes les barres-mères produites (et donc les poutrelles issues) auront cette dimension commune. Afin de ne pas plus entrer dans les détails de la fabrication, nous accepterons ceci sans plus de commentaires.

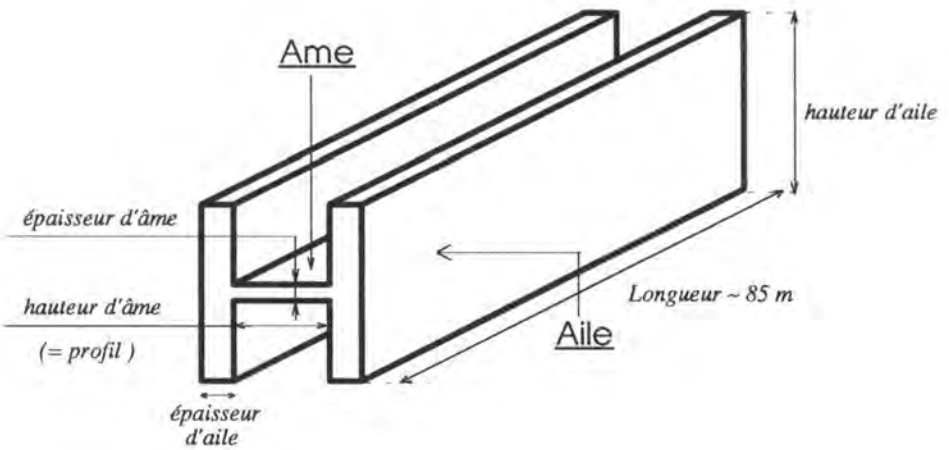
Nous sommes maintenant prêts pour aborder la nomenclature.

B. Nomenclature.

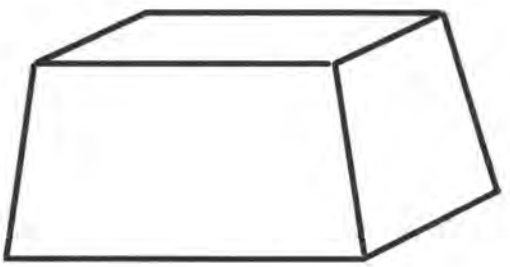
Poutrelle : Produit fini provenant de la découpe transversale d'une barre-mère.



Barre-Mère (BM) : "immense poutrelle", de l'ordre de 85 mètres.



Lingot : Bloc d'acier produisant une barre-mère par laminage.



Nuance : qualité d'acier.

Profil (d'une BM, d'une poutrelle) : Hauteur d'âme.

La notion de profil correspond à la dimension de barre-mère (de poutrelle) qui est fixée et invariable durant une session de laminage de huit heures.

Dérivé : triplet des mesures (épaisseur d'âme, hauteur d'aile, épaisseur d'aile).

La notion de dérivé correspond aux dimensions variables d'une poutrelle.
La longueur n'intervient qu'à la découpe de la barre-mère et le profil est fixé.

Une BM est caractérisée par :

- son profil,
- son dérivé,
- sa longueur,
- sa nuance.

Poste :

{ poutrelles | même client, même profil, même nuance, même dérivé, même longueur }

Informellement, un poste est l'ensemble des poutrelles strictement identiques commandées par un même client. Une commande de client peut regrouper plusieurs postes.

Extrait (de commandes):

{ postes | même profil }

Informellement, un extrait (de commandes) est l'ensemble des postes dont les poutrelles ont même profil (c.-à-d. même hauteur d'âme). Les clients, longueurs, dérivés et nuances sont confondus.

Un extrait (de commandes) représente une session de travail de huit heures avec un même jeu de cylindres de laminage.

C. Des commandes de poutrelles à la planification de la production.

1. Les grandes étapes.

Nous allons aborder maintenant la planification lors de la production des poutrelles. Elle est réalisée en trois étapes, chacune de ces étapes correspondant en fait à une application particulière :

- la Planification de l'Extrait de commande
- la Planification de la Production de l'Extrait de commande
- la Production de l'Extrait de commande

Voyons plus précisément de quoi il s'agit. Pour cela, jetons un oeil, à titre d'exemple, sur l'ensemble de toutes les commandes supposées connues actuellement :

N° de commande	Client	Destination	Nombre de poutrelles	Ensemble des Dimensions (Profil,Dérivé, Longueur)	Qualité d'acier (nuance)
1	C1	L1	N1	D1	Q1
1	C1	L1	N2	D2	Q1
1	C1	L2	N3	D3	Q2
2	C2	L2	N4	D5	Q2
2	C2	L3	N5	D4	Q1
3	C3	L1	N6	D1	Q1
4	C1	L1	N7	D5	Q2
4	C1	L4	N8	D5	Q3
5	C4	L2	N9	D5	Q3

← *un poste*

Ensemble exhaustif des commandes

Pratiquement, les commandes seront traitées par groupes de postes. Un groupe de postes de commandes sera appelé *extrait de commandes*. Dans notre exemple, un extrait de commandes (par abus de langage, nous disons *extrait*) correspond à un ensemble de lignes du tableau, non nécessairement consécutives ni ordonnées. Choisir les postes entrant dans un extrait de commandes, c'est *planifier un extrait de commandes*. Ceci fait l'objet de la première application. Une fois qu'un extrait de commandes à réaliser est planifié, il peut être produit. Mais la production d'un extrait ne se fait pas selon un ordre aléatoire des postes. Elle va donc elle-même aussi être planifiée. Nous parlerons alors de la *planification de la production de l'extrait*. Planifier la production d'un extrait, c'est décider de l'ordre dans

lequel les postes de l'extrait seront produits. L'objet de la seconde application est ainsi déterminé. Il ne reste plus maintenant qu'à produire l'extrait dans l'ordre des postes. C'est là l'objet de la dernière application, également appelée *Commande d'Acier*.

Très précisément, c'est l'étude de l'optimisation de l'application *Commande d'Acier* qui fait l'objet de ce document.

Il existe en réalité une étape supplémentaire faisant également l'objet d'une application : *les Ordres de Chargement*. Cette étape consiste à déterminer quels seront les wagons nécessaires à l'expédition de l'extrait de commandes, à commander ces wagons et à les ordonnancer. Remarquons que l'application des Ordres de Chargement peut être antérieure à la planification de la production de l'extrait. Cela permet en effet de favoriser l'optimisation du chargement et des expéditions. Il en est ainsi à l'aciérie d'Esch-Belval.

Pour permettre la production des postes, quatre applications vont donc se succéder :

1. *Planification de l'Extrait,*
2. *Ordres de Chargement,*
3. *Ordonnancement des Postes,*
4. *Commande d'Acier.*

Cependant, ces applications sont toutes interdépendantes. Nous allons vite nous rendre compte que nous sommes plongés dans un monde où l'idéal n'existe pas et fait place en réalité à un compromis arbitraire d'idéaux. Optimiser chacune de ces applications individuellement ne revient pas à optimiser toute la chaîne de production des poutrelles d'acier, bien au contraire. Analysons donc plus particulièrement chacune de ces applications pour en comprendre les interactions.

2. De la Planification de l'Extrait à la Commande d'Acier.

La planification de l'extrait

Il est intéressant de se demander quels pourraient être les critères de planification de l'extrait, c'est-à-dire les critères de sélection des postes entrant dans le prochain extrait. En voici quelques-uns, assez intuitifs :

- sélectionner les commandes selon la date.

  objectif: premier client, premier servi.

- sélectionner les commandes en fonction des délais de livraison.

  objectif: respect des obligations.

- sélectionner les commandes en fonction du lieu de destination.

  objectif: minimiser les trajets d'expédition.

- sélectionner les commandes selon les qualités d'acier.

  objectif: favoriser la fabrication.

- sélectionner les commandes en fonction des dimensions des poutrelles.

  objectif: favoriser la fabrication, minimiser le nombre de wagons d'expédition.

- sélectionner les commandes en fonction du stock de poutrelles.

  objectif: vider le stock.

- etc...

Il n'est pas nécessaire, dans le contexte de ce document, de connaître les critères réels permettant la planification des extraits de commandes. Nous travaillerons toujours sur base d'un extrait donné. Il est cependant utile de rappeler deux caractéristiques de tout extrait de commandes:

- dans un extrait de commandes, toutes les poutrelles ont une de leurs dimensions commune : la hauteur d'âme, appelée profil (hauteur de la partie centrale de la poutrelle, v. nomenclature).
- un extrait correspond à une session de travail de 8 heures environ.

Voici à titre d'exemple un extrait de commandes qui, comme son nom l'indique, est un sous-ensemble de l'ensemble exhaustif des commandes restant à honorer :

N° de commande	Client	Destination	Nombre de poutrelles	Ensemble des Dimensions (Profil,Dérivé, Longueur)	Qualité d'acier (Nuance)
1	C1	L2	N3	D3	Q2
2	C2	L2	N4	D5	Q2
4	C1	L1	N7	D5	Q2
4	C1	L4	N8	D5	Q3
5	C4	L2	N9	D5	Q3

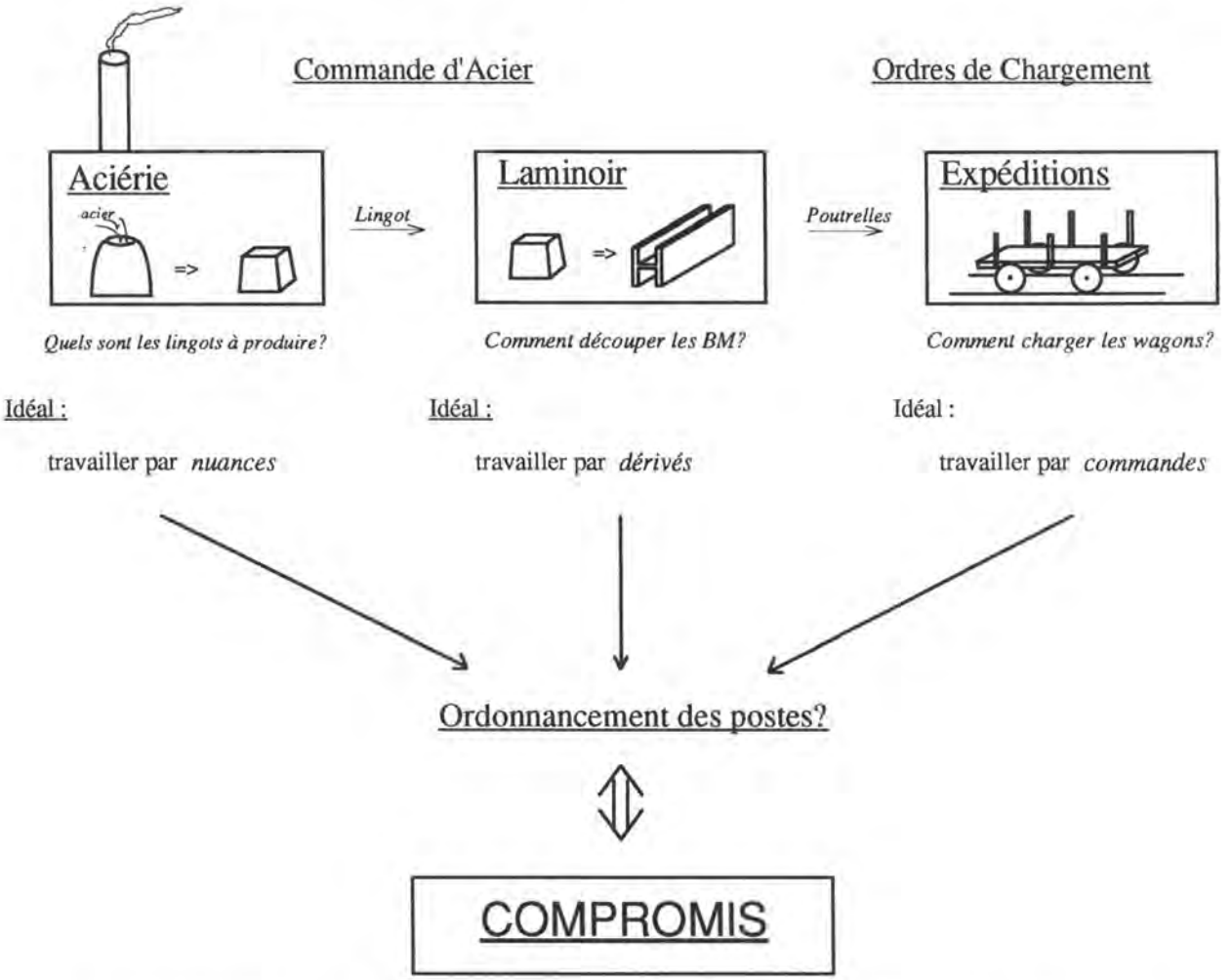
Un extrait de commandes

Les ordres de chargement et l'ordonnancement des postes

La planification de l'extrait étant réalisée, nous pouvons nous atteler à la planification de sa production et aux ordres de chargement. Rappelons qu'il a été décidé de favoriser les ordres de chargements et l'ordonnancement des wagons d'expédition en exécutant d'abord l'application des Ordres de Chargement. Nous pouvons en fait considérer que l'application des Ordres de Chargement effectue notamment un préordonnancement des postes. En d'autres mots, dès que l'extrait de commandes est planifié, ses postes peuvent être ordonnancés pour être produits par la suite. C'est donc à l'ordonnancement des postes de l'extrait que nous nous intéressons maintenant, en tenant compte de la priorité des contraintes d'expédition. Pour ordonnancer les postes, de nombreux critères peuvent également être décisifs, dont ceux de la planification de l'extrait, cités ci-dessus.

L'ordonnancement des postes lors de la planification de production d'un extrait est rendue difficile par le fait que l'idéal de production est différent à chacune des étapes de la fabrication. Il semble donc utopique de vouloir réaliser un ordonnancement optimal au sein d'un extrait de laminage. L'ordonnancement ne sera raisonnablement optimal qu'au sens de compromis réalisés tout au long du processus de fabrication. Le schéma suivant permet de percevoir cela :

PLANIFICATION DE LA PRODUCTION D'UN EXTRAIT :



La planification de la production d'un extrait ne peut en réalité qu'être réalisée selon un compromis d'idéaux:

- Aciérie : "ne pas entamer plus de n nuances",
- Laminoir : "ne pas entamer plus de n dérivés",
- Expéditions : "ne pas entamer plus de n destinations".

Ce compromis d'idéaux détermine l'ordonnancement de l'extrait planifié, mais aussi la planification elle-même de l'extrait, c'est-à-dire le choix des postes contenus dans l'extrait à produire. Ce problème ne nous concerne pas en soi. Supposons donc que nous ayons un extrait planifié et ordonnancé; cet extrait étant planifié et ordonnancé, sa production pourra (enfin) être lancée.

N° de commande	Client	Destination	Nombre de poutrelles	Ensemble des Dimensions (Dérivé,Profil ,Longueur)	Qualité d'acier (nuance)
4	C1	L4	N8	D5	Q3
5	C4	L2	N9	D5	Q3
4	C1	L1	N7	D5	Q2
2	C2	L2	N4	D5	Q2
1	C1	L2	N3	D3	Q2

Extrait ordonnancé selon un compromis d'idéaux

La Commande d'Acier

L'extrait planifié est maintenant ordonnancé selon un compromis arbitraire de chacun des idéaux rencontrés dans les différentes étapes de la production. Nous savons donc quels postes produire et dans quel ordre. Eh bien produisons-les! Ceci représente la tâche qui m'a été confiée, sous la direction de Marc Lescrenier :

Déterminer la séquence des lingots à produire ainsi que la découpe des barres-mères en poutrelles, de façon à recouvrir l'extrait avec un minimum d'acier, en fonction du processus de fabrication et en respectant l'ordonnancement des postes (indépendamment des contraintes de chargement, car prises en compte dans le pré-ordonnancement des postes).

D. La Commande d'Acier.

1. Ordonnancement des poutrelles.

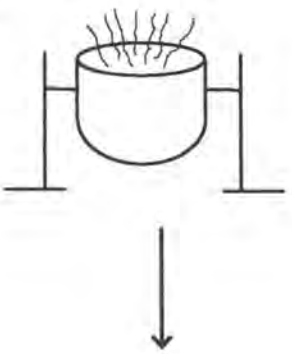
L'étude de la Commande d'Acier est en soit très simple. Nous verrons cependant qu'il y a là matière à réflexions. Résumée en quelques mots, la Commande d'Acier pour un extrait donné ordonnancé consiste à établir une séquence dans les poutrelles à fabriquer, c.-à-d. un plan de découpe des barres-mères, en respectant dans une certaine mesure l'ordre établi dans les postes de l'extrait par les applications précédentes. Le but est de minimiser la quantité d'acier nécessaire à la fabrication des poutrelles, toute contrainte étant par ailleurs considérée. En effet, cette quantité d'acier est fortement liée à l'ordre de fabrication des poutrelles, ceci étant principalement dû au processus de fabrication. Les poutrelles sont fabriquées à partir de la découpe de barres-mères, qui ne sont rien d'autre que d'immenses poutrelles (de l'ordre de 85 mètres). Ces barres-mères sont, elles, produites par laminage de lingots d'acier. Enfin, les lingots d'acier sont, eux, coulés dans des lingotières de différents types.

Outre le séquencement des poutrelles, c'est-à-dire le plan de découpe des barres-mères, il faudra également au sein de l'application Commande d'Acier déterminer l'affectation des lingots, dont sont issues les barres-mères, dans les lingotières.

2. Description du processus de fabrication.

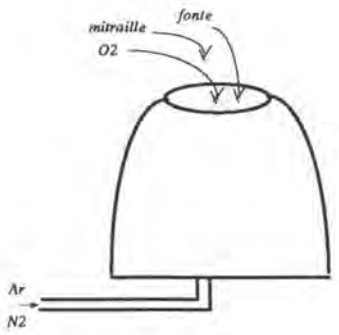
Voyons d'un peu plus près le processus de fabrication des poutrelles d'acier. A cette fin, troquons notre crayon contre un casque de sécurité et partons visiter l'usine d'Esch-Belval, aciérie du groupe ARBED Luxembourg.

Haut-Fourneau



Production de la fonte

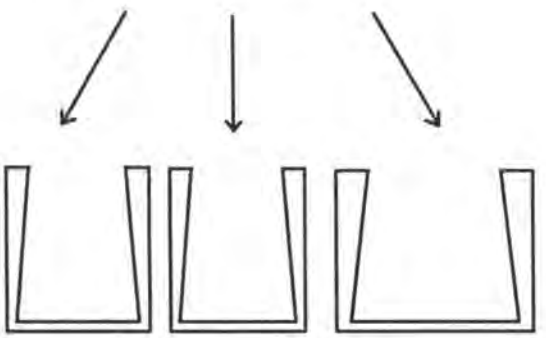
Aciérie



Production de l'acier brut

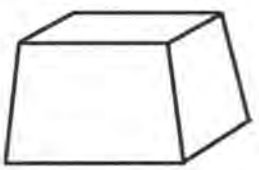
Traitement de l'acier brut

Production d'une nuance



Coulage des lingots pour cette nuance

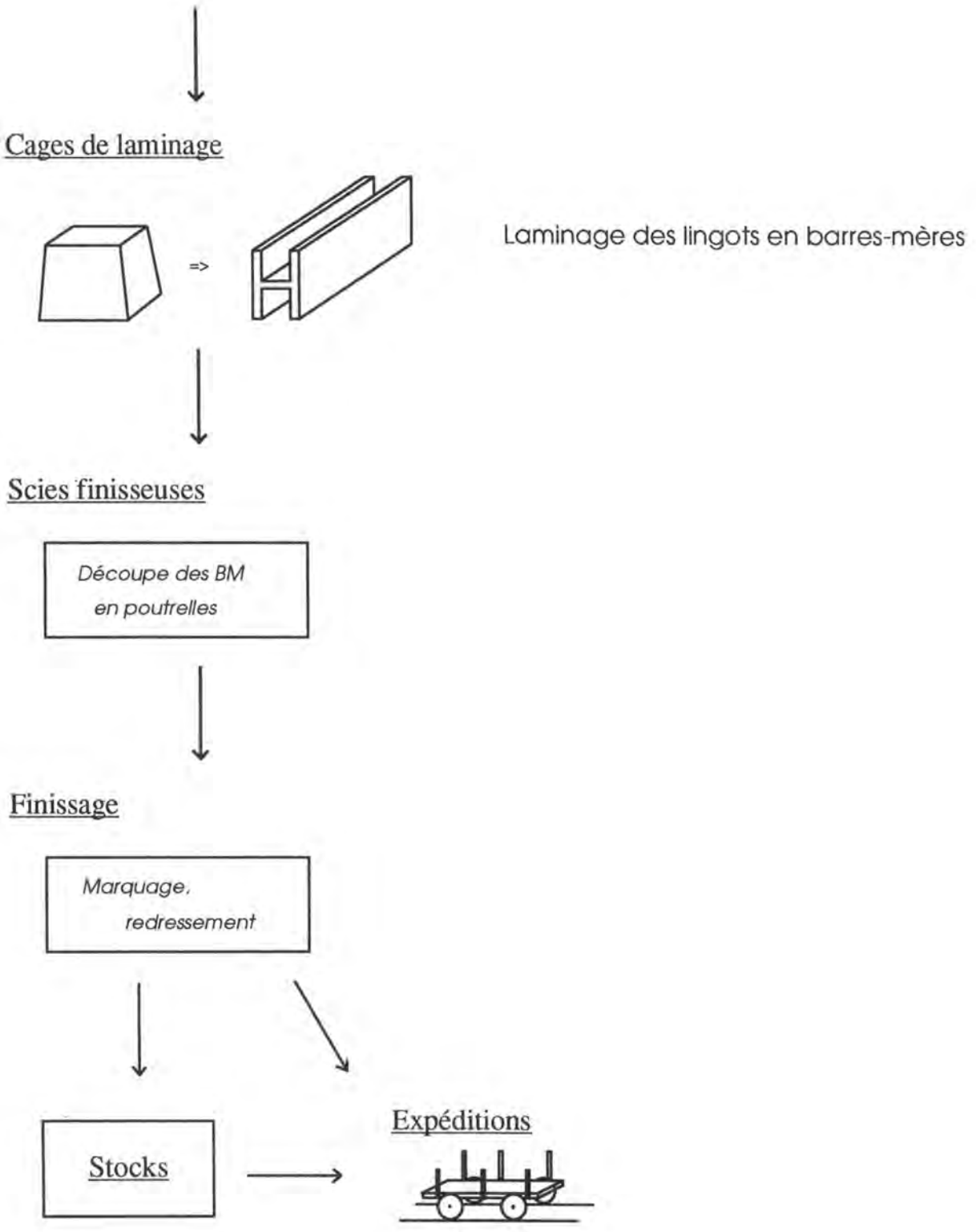
lingotières



Démoulage des lingots

Fours Pits ← Stocks

Maintient des lingots à T°



La commande d'acier s'étend des lingotières aux scies finisseuses.

3. L'unité de base de la Commande d'Acier.

Nous connaissons maintenant le processus de fabrication des poutrelles, ainsi que le traitement attendu au sein de la Commande d'Acier. Regardons à présent quelle sera l'unité de base de traitement de cette application. A cet effet, reprenons notre exemple, sous la forme d'un tableau, d'un extrait planifié supposé donné ordonnancé :

N° de poste	N° de commande	Client	Destination	Nombre de poutrelles	Ensemble des Dimensions (Dérivé,Profil ,Longueur)	Qualité d'acier (nuance)
1	4	C1	L4	N8	D5	Q3
2	5	C4	L2	N9	D5	Q3
3	4	C1	L1	N7	D5	Q2
4	2	C2	L2	N4	D5	Q2
5	1	C1	L2	N3	D3	Q2

Extrait ordonnancé selon un compromis d'idéaux

Chaque ligne du tableau est un poste de l'extrait. Les numéros de ligne du tableau reflètent l'ordonnancement des postes, établi précédemment à la Commande d'Acier.

Au vu de l'ordre dans lequel les quatre applications de la planification de production des postes d'un extrait se succèdent, l'optimisation de la Commande d'Acier s'effectue au niveau d'un extrait de laminage dont les postes ont été triés par l'application d'Ordonnancement des Postes, en fonction des Ordres de Chargement. La Commande d'Acier, elle, a pour objet d'ordonnancer les poutrelles de chacun des postes de l'extrait afin d'optimiser la quantité d'acier nécessaire à leur fabrication. Les critères influants dans l'ordonnancement de la Commande d'Acier seront, eux, de nature purement matérielle et technique : le processus de fabrication montre que seules des poutrelles strictement identiques dans leurs dimensions, à l'exception de la longueur, et strictement identiques dans leur qualité d'acier peuvent être fabriquées au sein d'une même barre-mère. La Commande d'Acier doit donc travailler au niveau de groupes de postes de même nuance et dimensions, exception faite de la longueur. Pour cette raison, l'ensemble de toutes les dimensions d'une poutrelle, sauf sa longueur et le profil (fixé au sein de l'extrait), est appelé *dérivé* d'une poutrelle. La qualité d'acier est appelée *nuance*. Un groupe de postes au sens de la Commande d'Acier sera identifié par le couple de données (dérivé,nuance). Puisqu'il ne faut pas "trop" perturber l'ordonnancement de ces postes effectué par les applications précédant la Commande d'Acier, ce sont donc des postes consécutifs dans l'extrait ordonnancé qui seront regroupés. L'objet de traitement de la Commande d'Acier est ainsi établi : il s'agit des couples (dérivé,nuance) de l'extrait ordonnancé. Un couple (dérivé,nuance) comprend un ou plusieurs postes consécutifs dans l'extrait, dont le dérivé et la nuance sont identiques.

N° de poste	N° de commande	Client	Destination	Nombre de poutrelles	**Dérivé**	Longueur	**Nuance**
1	4	C1	L4	N8	D'5	Lg1	Q3
2	5	C4	L2	N9	D'5	Lg2	Q3
3	4	C1	L1	N7	D'5	Lg2	Q2
4	2	C2	L2	N4	D'5	Lg4	Q2
5	1	C1	L2	N3	D'3	Lg3	Q2

Extrait ordonnancé découpé en couples (dérivé,nuance)

Toute liberté sera laissée à la Commande d'Acier au sein d'un couple (dérivé,nuance). En particulier, les poutrelles des différents postes d'un même couple (dérivé,nuance) pourront être mélangées. L'acceptation d'introduire ce désordre permet une meilleure optimisation dans la fabrication des poutrelles.

Aux yeux de la Commande d'Acier, il n'est pas nécessaire de se charger des données concernant le numéro de poste, le numéro de commande, le client et la destination. Seules les informations concernant les couples (dérivé,nuance) à produire sont intéressantes. L'extrait à produire sera donc vu par la Commande d'Acier sous une forme particulière se réduisant à la considération de ces seules données :

Dérivé	**Nuance**	**Longueur**	**Nombre de poutrelles**
D'5	Q3	Lg1	N8
		Lg2	N9
D'5	Q2	Lg2	N7
		Lg4	N4
D'3	Q2	Lg3	N3

Extrait sous le regard de la Commande d'Acier

Le but de l'application Commande d'Acier est de déterminer la séquence optimale des barres-mères à fabriquer, ainsi que leurs combinaisons de découpe en poutrelles, pour produire un à un les couples (dérivé,nuance) de l'extrait donné. Ces couples étant produits un à un, nous allons donc concevoir une application travaillant au niveau d'un seul couple (dérivé,nuance). La Commande d'Acier sera ainsi réduite à un appel de notre application au sein d'une itération sur chaque couple de l'extrait. Ceci ne représente pas en soi qu'un bidouillage d'informaticien, car un avantage majeur se dégage : la gestion du stock se résume à déduire du total à produire le nombre de poutrelles présentes dans le stock, pour chacune des longueurs des couples (dérivé,nuance) traités. Cette mise à jour du nombre de poutrelles à produire se fera au sein de l'itération, car la production des poutrelles d'un couple donné peut entraîner la production d'un surplus de poutrelles. Le stock pourra donc aisément être mis à jour après chaque production d'un couple (dérivé,nuance).

4. Les contraintes à respecter.

L'application de la Commande d'Acier est en soi assez simple puisqu'il s'agit de placer des poutrelles dans des barres-mères, qu'à chaque barre-mère correspond un lingot, et qu'il ne reste alors plus qu'à placer les lingots dans les lingotières appropriées.

Différentes contraintes jettent cependant le trouble. Ces contraintes trouvent principalement leur origine dans le processus de fabrication :

a) La longueur de la barre-mère :

Les barres-mères voient leurs longueurs minimale et maximale spécifiées pour chaque dérivé de chaque profil.

Plus la taille de la barre-mère à laminer sera proche de la taille maximale, plus la productivité sera grande.

b) Les combinaisons de longueurs incompatibles :

Les contraintes portant sur les combinaisons sont dues à des contraintes physiques concernant les scies. Ces contraintes sont données ici sans plus de précision.

Si une barre-mère comporte des longueurs inférieures à 9 mètres, alors le nombre de découpes au sein de cette barre-mère doit être pair.

Si le nombre de poutrelles au sein d'une barre-mère est impair et si le nombre de poutrelles de longueur inférieure à 9 mètres au sein de cette barre-mère est supérieur au nombre de poutrelles de longueurs supérieures à 9 mètres, alors la découpe est impossible.

Une barre-mère peut comporter 13 poutrelles au maximum. Néanmoins, il est conseillé de limiter le nombre de découpes dans une barre-mère. En effet chaque découpe durant 15 secondes, il est préférable de ne pas dépasser 6 découpes afin que la scie ne devienne pas une ressource critique pour la production. Les objectifs fixés sont la production d'une barre-mère toutes les 90 secondes.

Pour une barre-mère donnée, il ne faut pas mélanger des poutrelles de postes distants de plus d'une certaine valeur (définie comme paramètre) au sein d'un couple (dérivé,nuance).

c) L'uniformisation des poids lingots :

Un poids lingot est le poids d'un lingot considéré, c.-à-d. le poids du lingot à partir duquel une barre-mère considérée est laminée. Une faible disparité des poids lingots est admise au sein d'un couple (dérivé,nuance). En d'autres termes, les lingots commandés pour la production d'un couple(dérivé,nuance) donné doivent être au plus de trois poids différents (paramètre). Les poids lingots sont arrondis à 50 kg près.

d) La gestion des sous-longueurs :

Les lingots reçus sont normalement répartis autours des lingots commandés (c.-à-d. selon une distribution respectant la loi normale). On peut considérer qu'en moyenne un poids lingot commandé sera reçu une fois sur deux plus petit de trois pour-cents. Il en est donc de même pour les barres-mères obtenues après laminage des lingots. Cette particularité est appelée *le pourcentage statistique fixe de perte d'acier par rapport à la commande.*

Afin de gérer la perte statistique fixe d'acier par rapport à la commande, la notion de sous-longueur est utilisée: des combinaisons de découpe pour les barres-mères reçues plus courtes sont prévues, de façon à recouvrir entièrement les poutrelles du couple(dérivé,nuance) traité lors de la découpe effective.

Exemple:

Sont à produire les poutrelles suivantes : 10×15 m, 20×6 m, 15×10 m.

Les barres-mères étant de 85 mètres au plus, une combinaison de découpe nous intéressant pourrait être :

$$2 \times 15 \text{ m} + 4 \times 6 \text{ m} + 3 \times 10 \text{ m} = 84 \text{ m}$$

Cette combinaison nous intéresse particulièrement car, utilisée cinq fois, elle permet de produire tout le couple.

Commandé : 5×84 m

Reçu, à cause du pourcentage fixe de perte : 2×84 m et 3×81 m

Découpe :	Reste à produire :
$2 \times$ 84 m : selon $\{2 \times 15 \text{ m} + 4 \times 6 \text{ m} + 3 \times 10 \text{ m}\}$	6×15m, 12×6m, 9×10m
$2 \times$ 81 m : selon $\{3 \times 15 \text{ m} + 6 \times 6 \text{ m} = 81 \text{ m}\}$	9×10m
$1 \times$ 81 m : selon $\{8 \times 10 \text{ m} = 80 \text{ m}\}$	1×10m

Les barres-mères reçues plus courtes sont appelées *sous-longueurs*, et leurs combinaisons de découpe, *combinaisons de découpe alternatives.*

Bien que judicieusement choisies, les combinaisons de découpe alternatives ne permettent pas la production de tout le couple, et un excès de 1 m reste inutilisé. La commande de $5 \times$ 84 m ne convient donc pas.

UNE SOLUTION :

Commande : $4 \times \{2 \times 15 \text{ m} + 4 \times 6 \text{ m} + 3 \times 10 \text{ m} = 84 \text{ m}\}$ et $2 \times \{5 \times 10 \text{ m} = 50 \text{ m}\}$

Reçu : 2×84m et 2×81m et 1×50m et 1× 48,5m

Découpe :	Reste à produire :
$2 \times$ 84 m : selon $\{2 \times 15 \text{ m} + 4 \times 6 \text{ m} + 3 \times 10 \text{ m} = 84\text{m}\}$	6×15m, 12×6m, 9×10m
$2 \times$ 81 m : selon $\{3 \times 15 \text{ m} + 6 \times 6 \text{ m} = 81 \text{ m}\}$	9×10m
$1 \times$ 50 m : selon $\{5 \times 10 \text{ m} = 50 \text{ m}\}$	4×10m
$1 \times$ 48,5 m : selon $\{4 \times 10 \text{ m} = 40 \text{ m}\}$	rien

Les combinaisons et combinaisons alternatives de découpe permettront cette fois la production de tout le couple. Cette solution pourra donc être retenue. Il y a cependant production d'un excès de 8,5m inutilisé.

e) L'ajout du pourcentage statistique fixe :

Une solution conjointe à la prévision de combinaisons alternatives de découpe pour les barres-mères reçues plus courtes du pourcentage statistique fixe de perte d'acier est de commander ces barres-mères augmentées du pourcentage fixe de perte.

Exemple:

Dans l'exemple ci-dessus, il serait plus intéressant de commander :

$5\times \{ (2 \times 15\ m + 4 \times 6\ m + 3 \times 10\ m)\ *103\% = 87m\}$

Si cette découpe est possible (barres-mères > 85m !), il résultera de leur découpe un excès moins important restant inutilisé:

En effet: reçu $2\times 87m$ et $3\times 84m$

excès produit : $3\times (87m - 84m) = 6\ m.$

Exemple:

Soient à produire les poutrelles suivantes : 20×18 m.

Les barres-mères étant de 85 m au plus, une combinaison de découpe pourrait être :

$4 \times 18\ m = 72\ m$

Une combinaison de découpe alternative pour la sous-longueur ne peut qu'être :

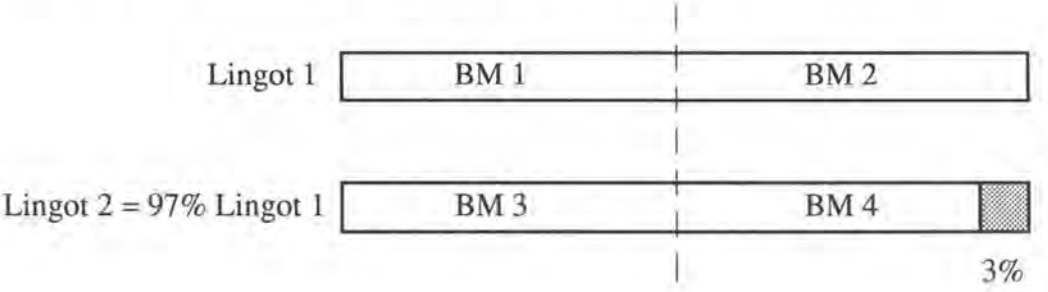
$3 \times 18\ m = 54\ m$

En commandant 6 barres-mères de 72 m, il sera possible de produire tout le couple avec une poutrelle en surplus et une perte importante d'acier.

En commandant 5 barres-mères de 1.03 * 72 m = 75 m, la quantité d'acier utilisée est très nettement réduite.

f) Les deux ébauches :

Lorsque deux barres-mères sont laminées à partir d'un lingot, ce lingot est dit *à deux ébauches* . Le pourcentage statistique fixe de perte d'acier par rapport à la commande ne porte alors que sur une barre-mère pour quatre au lieu d'une barre-mère sur deux.



g) Le respect de l'ordonnancement des postes :

L'ordre des postes au sein des couples(dérivé,nuance) devra être respecté dans une certaine mesure. Il est clair que ce respect favorisera le chargement des expéditions, mais qu'il aura tendance à détériorer dans des proportions non négligeables le rendement et la productivité au laminoir.

h) La gestion des lingotières :

Les lingots sont coulés dans des lingotières de différents types caractérisés par des niveaux minimum et maximum de remplissage. Le nombre de lingotières disponibles pour chacun des types varie. Il faut s'assurer que les lingots commandés peuvent tous être coulés dans des lingotières disponibles.

i) Les surplus de laminage :

Si les chutes donnent lieu à des longueurs de poutrelles précisées en fonction de leur dérivé, elles sont dites vendables et peuvent être stockées. Dans le cas contraire il s'agit de mitraille non récupérable. Des poutrelles provenant de l'extrait peuvent être produites en supplément.

Cette contrainte ne sera pas considérée dans la modélisation. Chaque surplus d'acier produit sera considéré comme mitraille. Le résultat de la modélisation pourra donc être reconsidéré plus favorablement.

j) La contrainte de temps :

L'extrait ordonnancé doit être produit endéans l'heure.

E. Optimisation de la Commande d'Acier.

1. L'heuristique.

Une première solution envisagée pour traiter le problème de la Commande d'Acier est une solution itérative de traitement des poutrelles, une barre-mère à la fois. Connaissant le nombre total de poutrelles à produire pour les différentes longueurs d'un couple (dérivé, nuance), une barre-mère est constituée de façon à respecter 'au mieux' les contraintes. Les poutrelles "placées" au sein de la barre-mère constituée sont alors "gommées" de l'ensemble des poutrelles à produire. Ce processus est réitéré jusqu'à ce que toutes les poutrelles aient été "gommées". Le problème réside dans le choix des poutrelles à placer dans la barre-mère constituée, car ce choix ne peut mettre de contrainte en défaut. Cette solution est appelée solution heuristique.

L'inconvénient majeur de cette solution par heuristique est que, de toute évidence, elle se restreint à une vue locale du problème par le fait qu'elle est purement itérative, et que la constitution des dernières barres-mères est conditionnée par la constitution des précédentes. A chaque itération de la solution heuristique, il y a diminution des libertés.

2. Le Programme Linéaire.

Il serait donc intéressant de pouvoir traiter le problème non pas de façon itérative, c'est-à-dire selon une vue locale, mais bien de façon globale, c'est-à-dire en constituant simultanément toutes les barres-mères au sein desquelles seront placées toutes les poutrelles à produire. Ici encore, il faut s'assurer que toutes les contraintes sont vérifiées.

C'est dans cette optique que l'on songe à envisager une solution par le biais de la théorie mathématique de la programmation linéaire.

Ceci étant dit, mettons-nous au travail! L'objet de ce document est enfin cerné avec précision : résoudre le problème de la Commande d'Acier par le biais de la théorie mathématique de la programmation linéaire.

3. L'Application Globale.

Il serait cependant utopique de croire qu'un programme linéaire représente la solution miracle pour résoudre entièrement le problème de la Commande d'Acier. Nous allons effectivement être confronté rapidement à une complexité gênante qui augmente de façon exponentielle le temps de calcul. C'est pourquoi nous allons en réalité concevoir une application globale dont le coeur sera un programme linéaire. L'application globale sera définie comme suit :

- prétraitement de la Commande d'Acier,

- Programme Linéaire,
- interprétation des résultats du P.L.,
- post-traitement de la Commande d'Acier.

Les pré- et post- traitements seront destinés à réduire le nombre de variables du programme linéaire.

Mise en évidence de la complexité : une première ébauche du P.L.

Etant donné un couple (dérivé, nuance) donné quelconque contenant K postes, simplifions le programme linéaire au seul calcul du plan de découpe des barres-mères en poutrelles, avec pour seule contrainte la production de tout le couple. Nous allons donc uniquement établir des combinaisons de poutrelles qui constitueront les barres-mères, en s'assurant que le couple entier pourra ainsi être produit sur base des combinaisons de découpe seules.

Une combinaison de découpe de barre-mère sera un K-uplet de coefficients indiquant le nombre de poutrelles de chacun des postes présentes dans la barre-mère.

exemple : soit un couple (dérivé,nuance) contenant 4 postes.
on a l_1, l_2, l_3, l_4 la longueur de poutrelle de chacun des postes,
n_1, n_2, n_3, n_4 le nombre de poutrelles dans chacun des postes.
si une barre-mère est constituée de $2xl_2+ l_3$,
sa combinaison de découpe sera le 4-uplet (0,2,1,0)

Essai de modélisation d'un Programme Linéaire :

Données :

K ≡ nombre de postes dans le couple traité.

k ≡: indice sur les postes du couple, k=1..K.

l_k ≡ longueur des poutrelles du poste k, k=1..K.

n_k ≡ nombre de poutrelles dans le poste k, k=1..K.

Variables : Calculer les combinaisons de découpe des barres-mères et leur nombre d'utilisation

I ≡ nombre de combinaisons calculées.

i ≡ indice sur les combinaisons, i=1..I

(Ci,k)k ≡ $i^{ème}$ combinaison, k=1..K, i=1..I

≡ nombre de poutrelles l_k présentes dans la combinaison i

$N_i \equiv$ nombre de barres-mères découpées selon la combinaison $(C_{i,k})_k$.

Formalisation :

$$\text{minimiser}\left(\sum_{i=1}^{I}\left(N_i\sum_{k=1}^{K}(C_{i,k}*l_k)\right)\right) \qquad \{\text{ minimiser la longueur totale des BM}\}$$

$$\text{avec} \quad \forall i = 1..I: \sum_{k=1}^{K}(C_{i,k}*l_k) \le 85 \qquad \{\text{ longueur max. d'une barre-mère }\}$$

$$\forall i = 1..I: \sum_{k=1}^{K}(C_{i,k}*l_k) \ge 60 \qquad \{\text{ longueur min. d'une barre-mère }\}$$

$$\forall k = 1..K: \sum_{i=1}^{I}(C_{i,k}*N_i) \le n_k \qquad \{\text{ recouvrement du couple }\}$$

Nous nous heurtons déjà à plusieurs problèmes de modélisation :

- I ne peut pas être une variable mais doit être une donnée.
- la modélisation est quadratique et non linéaire.

Pour contourner le premier problème, il suffit de donner I en donnée, avec pour valeur le nombre maximum de combinaisons de découpe admises. Puisque le nombre d'utilisation de chaque combinaison est calculé, en particulier des combinaisons peuvent ne pas être utilisées.

Plus gênant est le problème de non-linéarité. La solution est évidemment de ne pas calculer les N_i. Dans ce cas, une combinaison devant être utilisée N_i fois apparaîtra N_i fois. Cette solution requiert beaucoup trop de variables. Nous devons donc la rejeter. De plus, l'expérience montre qu'une même combinaison de découpe de barres-mères (c'est-à-dire une affectation de poutrelles dans les barres-mères) sera en général utilisée un grand nombre de fois. Cela est une caractéristique que l'on privilégiera d'ailleurs pour favoriser la découpe en série des barres-mères.

Le calcul de l'ensemble des combinaisons de découpe valides (c'est-à-dire l'affectation de poutrelles dans des barres-mères) peut être effectué en dehors du programme linéaire. Le programme linéaire n'aura alors qu'à choisir parmi toutes les combinaisons proposées celles qui sont les meilleures, et déterminer le nombre de fois que chacune d'elles sera utilisée. Nous allons donc uniquement calculer les N_i :

Programme Linéaire :

Données : K ≡ nombre de postes dans le couple traité.

k ≡ indice sur les postes du couple, k=1..K

l_k ≡ longueur des poutrelles du poste k

n_k ≡ nombre de poutrelles dans le poste k

I ≡ Nombre de combinaisons proposées; i=1..I

$(C_{i,k})_k$ ≡ nombre de poutrelles l_k présentes dans la combinaison i

Variables : N_i ≡ nombre de fois que la combinaison $(C_{i,k})_i$ est utilisée.

Fonction objective :

$$\text{minimiser}\left(\sum_{i=1}^{I}\left(N_i\sum_{k=1}^{K}(C_{i,k}*l_k)\right)\right) \quad \{\text{ minimiser la longueur totale des BM}\}$$

$$\text{avec} \quad \forall i=1..I:\sum_{k=1}^{K}(C_{i,k}*l_k)\le 85 \quad \{\text{ longueur max. d'une barre-mère }\}$$

$$\forall i=1..I:\sum_{k=1}^{K}(C_{i,k}*l_k)\ge 60 \quad \{\text{ taille minimale d'une barre-mère }\}$$

$$\forall k=1..K:\sum_{i=1}^{I}(C_{i,k}*N_i)\le n_k \quad \{\text{ recouvrement du couple }\}$$

Ceci est très avantageux du point de vue de la complexité du programme linéaire. Encore faut-il évidemment pouvoir calculer les combinaisons de découpe, ce qui ne pose pas de problème majeur. En outre, toutes les contraintes portant sur le rejet de combinaisons de découpe pourront être placées dans le module de génération des combinaisons, et donc en dehors du programme linéaire, ce qui réduira encore la complexité.

Prétraitement de la Commande d'acier :

Le calcul de l'ensemble des combinaisons valides de découpe de barres-mères (possibilités d'affectation de poutrelles au sein de barres-mères) constitue le premier module du prétraitement dans l'application globale. Une combinaison est considérée comme valide si elle satisfait à toutes les contraintes concernées par la découpe des barres-mères. La génération des combinaisons valides fera appel à un module d'élimination de combinaisons valides mais jugées peu intéressantes. Ceci sera explicité plus en détail plus tard.

Un autre module préalable à l'appel du programme linéaire est le module proposant l'ensemble des poids lingots à partir desquels les barres-mères pourront être laminées. Le

processus de fabrication montre qu'une barre-mère est équivalente à un lingot. Ce module proposera donc pour poids lingots l'ensemble des longueurs des combinaisons calculées. Ici encore, certaines contraintes portant sur les poids lingots pourront être vérifiées et diminuer ainsi la complexité du programme linéaire.

Remarques :

- bien qu'équivalents à la notion de barre-mère, la notion de lingot est indispensable car sujette à certaines contraintes.
- la notion de barre-mère correspond dans notre contexte à une combinaison de découpe.

Interprétation des résultats :

Le module d'interprétation des résultats quant à lui est destiné à rendre disponible sous forme de résultats interprétables les valeurs des variables calculées par le programme linéaire.

Post-traitement de la Commande d'Acier :

Le module de post-traitement dans l'application globale sera expliqué plus loin. Il consiste entre autres à la gestion des grands couples.

L'Application Globale :

Nous allons donc concevoir une application globale pour répondre au problème de la Commande d'Acier, telle que :

soit un couple (dérivé, nuance) donné :

- calcul et validation des combinaisons de découpe des barres-mères,
- calcul des poids lingots,
- programme linéaire : choix des barres-mères, affectation dans les lingots,...
- interprétation des résultats du programme linéaire,
- gestion interne.

III. Optimisation de la Commande d'Acier

A. La voie de la programmation linéaire.

1. Un programme linéaire au coeur d'une application globale.

Une solution au problème d'optimisation de la Commande d'Acier peut être envisagée par la voie de la théorie mathématique de la programmation linéaire. Bien que cette solution par programmation linéaire soit attrayante, elle ne pourra être envisagée aisément. Nous nous heurtons en réalité à un problème non linéaire d'une grande complexité. Elle mérite cependant d'être analysée en profondeur, car s'il est possible de réduire le problème à un problème linéaire, nous pourrons jouir d'un avantage majeur de la théorie de la programmation linéaire : l'optimalité de la solution trouvée.

Le problème peut se résumer à un problème d'affectations multiples avec contraintes : il s'agit de placer au mieux des poutrelles dans des barres-mères. Pour cela, nous allons établir des combinaisons de poutrelles. Chaque combinaison correspondra alors à une barre-mère réalisable. Mais ces barres-mères sont fabriquées à partir de lingots. Il faut donc déterminer le poids du lingot à partir duquel la barre-mère sera laminée. Nous dirons que nous cherchons à affecter les barres-mères dans les lingots. Une fois que l'on connaît les lingots à fabriquer, encore faut-il les affecter aux lingotières.

Concrètement, un programme linéaire sera le coeur d'une application globale répondant aux exigences du problème de l'optimisation de la Commande d'Acier. L'application globale quant à elle assurera la gestion du programme linéaire à plusieurs niveaux tels :

- la préparations des données utilisées par le P.L.
- l'extraction et l'interprétation des résultats fournis par le P.L. en une solution interprétable par l'utilisateur
- la gestion de couples de très grandes tailles.

La modélisation idéale du P.L. serait de pouvoir, par le P.L.:

1) établir les meilleurs arrangements de poutrelles au sein des barres-mères, de sorte que leur découpe ainsi planifiée permette la production de toutes les poutrelles demandées, sans surplus. Autrement dit, établir les meilleures combinaisons de découpe.

2) déterminer les lingots qui permettent le laminage des barres-mères constituées, de façon à n'avoir qu'un nombre limité de poids lingots (prise en compte de la contrainte d'uniformisation des poids lingots).

3) établir la meilleure affectation des lingots dans les lingotières disponibles, en s'assurant qu'aucun lingot n'est en attente d'une lingotière indisponible.

4) assurer que la solution trouvée vérifie toutes les contraintes spécifiées, et soit optimale.

Cet idéal relève de la programmation non linéaire.

Afin de réduire la complexité du problème et le transformer en un problème linéaire, le *calcul* de certaines variables va être remplacé par un *choix* du meilleur résultat parmi plusieurs proposés. Cela permet d'avoir moins de variables à calculer, car il ne reste plus qu'à choisir les meilleures valeurs parmi celles qui sont proposées.

Plus précisément, les points 1) et 2) ci-dessus deviennent:

1) choisir, parmi une liste proposée, les meilleures combinaisons de découpe des barres-mères, qui permettent la production de toutes les poutrelles demandées, avec un surplus minimum.

2) choisir, parmi une liste proposée, les poids lingots tels que pour chacun d'eux, le nombre de lingots calculé permette le laminage des barres-mères constituées en 1), de façon à n'avoir qu'un nombre limité de poids lingots.

Le point 3) sera limité à:

3) vérifier qu'une affectation des lingots dans les lingotières disponibles soit possible.

Le point 4), quant à lui, reste inchangé si ce n'est qu'il est peu probable d'avoir une solution optimale rapidement. Il est donc possible qu'il faille se contenter d'une solution qui approche la solution optimale, appelée solution intermédiaire.

Les pré- et post- traitements au P.L. :

Permettre au P.L. de choisir des combinaisons et des lingots parmi des listes suppose que celles-ci soient établies préalablement à son exécution.

Ceci fait l'objet de 2 modules indépendants du P.L. et gérés par l'application globale:

1) un module de calcul des combinaisons valides de longueur de poutrelles au sein de barres-mères: GENECOMB

2) un module de calcul de poids lingots valides: DOWEIGHT.

Le P.L. fournit une solution en termes de valeurs numériques relatives à chaque variable traitée. L'interprétation de ces valeurs numériques est réalisée par un 3ème module indépendant du P.L., également géré par l'application globale:

3) un module d'interprétation de la solution: SOLUTION.

2. Schéma de construction du P.L. au sein de l'application globale.

Nous sommes maintenant capables de modéliser une première ébauche du Programme Linéaire et de l'implanter au sein de l'application globale. Mais faisons preuve de modestie. Nous allons élaborer cela par étapes successives, en ne prenant en compte qu'une contrainte à la fois. A chaque étape de la modélisation, nous construirons et modifierons les modules de l'application globale. Puisque la première raison d'être de l'application globale est la diminution de la complexité du P.L., le mot "modélisation" sera aussi bien utilisé en parlant du P.L. que de l'Application Globale. Pour rappel, les contraintes à respecter sont spécifiées dans le chapitre II.D.4.

Rappelons que l'application travaille au niveau d'un couple(dérivé,nuance) donné :

Dérivé	Nuance	Longueur	Nombre de poutrelles
D'5	Q3	Lg1	N8
		Lg2	N9

Un couple (dérivé,nuance) traité

Puisque le couple (dérivé,nuance) est donné, le dérivé et la nuance sont fixés et nous pouvons les ignorer. Le couple (dérivé,nuance) traité est donc réduit à un ensemble de doublets $(Lg_k,N_k)_{k=1..K}$ des longueurs et nombres des poutrelles de chacun des K postes du couple (dérivé,nuance) donné. Par abus de langage, nous parlerons indifféremment de couple, couple (dérivé,nuance) et couples $(Lg_k,N_k)_{k=1..K}$ pour caractériser l'objet de traitement de toute l'application incluant P.L.

En dehors de la modélisation, nous avons besoin d'un module de génération des combinaisons et d'un module d'extraction des résultats du P.L. Considérons ces modules à notre disposition et commençons la modélisation du P.L. et de son application globale.

B. Le Programme Linéaire.

Le Programme Linéaire et les différents modules sous-jacents vont être élaborés en prenant en compte une contrainte à la fois. Pour rappel, les contraintes sont spécifiées dans le chapitre II.D.4.

1. Première ébauche : le recouvrement du couple.

Dans sa première ébauche, le Programme Linéaire sera limité au seul calcul du plan de découpe des barres-mères en poutrelles, avec pour seule contrainte la production de tout le couple. Nous allons donc uniquement établir des combinaisons de poutrelles qui constitueront les barres-mères, et nous assurer que le couple entier pourra ainsi être produit sur base des combinaisons de découpe seules. A ce stade, les contraintes a) et b) seront donc prises en compte.

Pour rappel, une combinaison de découpe de barre-mère sera un K-uplet de coefficients indiquant le nombre de poutrelles présentes de chacun des postes dans la barre-mère.

exemple : soit un couple (dérivé,nuance) contenant K=4 postes.
on a l_1, l_2, l_3, l_4 la longueur de poutrelle de chacun des postes,
n_1, n_2, n_3, n_4 le nombre de poutrelles dans chacun des postes.
si une barre-mère est constituée de $2xl_2+ l_3$,
sa combinaison de découpe sera le 4-uplet (0,2,1,0)

Conventions d'écriture :

les données sont positives,
N: un entier, R: un réel,
N* ou R*: tableau d'entiers ou de réels à une dimension,
N** ou R**: tableau d'entiers ou de réels à deux dimensions,
etc...

Les données :

- Le couple traité :

K :	nombre de postes du couple.	(N)
k=1..K :	indice sur les postes.	(N)
$Long_k$:	longueur des poutrelles du $k^{ème}$ poste du couple.	(R)
$NLong_k$:	nombre de poutrelles dans le $k^{ème}$ poste du couple.	(N)

Exemple :

Soit à produire le couple suivant : 10×15 m, 20×6 m, 15×10 m.

Nous avons alors pour données :

$K = 3$

$(Long_k) = (15,6,10)$

$(NLong_k) = (10,20,15)$

- Les combinaisons de longueur de poutrelles, établies par le module GENECOMB :

I : nombre de combinaisons établies (N)

i=1..I : indice sur les combinaisons (N)

$(Comb_{i,k})_{k=1..K}$: coefficients de la i^{ème} combinaison, c.-à-d. occurrences des $Long_k$ dans la i^{ème} combinaison.

(N**)

$LComb_i$: longueur de la combinaison i :

$LComb_i = |(Comb_{i,k})_{k=1..K}| = \Sigma_k \, Comb_{i,k} * Long_k$ (R)

Exemple :

Soit la i^{ème} combinaison proposée : 2×15 m + 4×6 m + 3×10 m = 84 m

Nous avons pour données :

$(Comb_{i,k})_k = (2,4,3)$

$LComb_i = 84$

Les variables du Programme Linéaire :

- Les nombres d'utilisation des combinaisons de découpe de barres-mères :

$NComb_i$, i = 1..I : nombre de fois que la combinaison $(Comb_{i,k})_{k=1..K}$ est utilisée

Modélisation du Programme Linéaire :

Toute contrainte étant par ailleurs considérée, la fonction objective consiste à minimiser la quantité d'acier totale utilisée :

$$\text{minimiser} \left(\sum_{i=1}^{I} NComb_i * LComb_i \right)$$

tel que :

$$\forall k = 1..K : \sum_{i=1}^{I} NComb_i * Comb_{i,k} \geq NLong_k \qquad \{ \text{ recouvrement du couple } \}$$

Modules sous-jacents :

GENECOMB :

Etablit les combinaisons valides de poutrelles du couple donné au sein d'une barre-mère. Les combinaisons non valides de longueurs des poutrelles sont rejetées lors de leur génération dans le module GENECOMB par appel à la procédure ELIM, décrite ultérieurement. La procédure ELIM regroupe tous les critères d'élimination des combinaisons cités dans l'énoncé des contraintes.

SOLUTION :

extrait les variables du P.L. en des résultats interprétables par :

$\forall$ $NComb_i \neq 0$, i=1..I : commander

$NComb_i \times$ la barre-mère $\{(Comb_{i,k} \times Long_k$ mètres $)_{k=1..K} = LComb_i$ mètres$\}$.

Première ébauche du Programme Linéaire :

Le modélisateur utilisé est XPRESS-MP de Dash Associates :

```
MODEL  CA_10            ! MODELISATION  1.0

TABLES
 DIM(2)  ! DIM(1) = Number of small beams different lengthes.
         ! DIM(2) = Number of valid combinaisons.

DISKDATA
  DIM(1) = DIM.XPR

TABLES
  LONG(DIM(1))          ! LONG(K) = K-th small beam length.
  NLONG(DIM(1))                 ! NLONG(K) = number of small beam of
                        !       K-th length to produce.
  COMB(DIM(2),DIM(1))   ! COMB(I,K) = number of small beams of
  L_COMB(DIM(2))                ! L_COMB(I) = Length of MB expected if made from
                        !               I-th combinaison.
VARIABLES
  NCOMB(DIM(2))                 ! NCOMB(I) = how many times combinaison I is used.

DISKDATA
  LONG(1) = LONG.XPR
  NLONG(1) = NLONG.XPR
  COMB(1,1) = COMB.XPR

ASSIGN
  L_COMB(I=1:DIM(2)) = SUM(K=1:DIM(1)) LONG(K) * COMB(I,K)
  NP = SUM(K=1:DIM(1)) NLONG(K)     ! Small beams total number to produce.

CONSTRAINTS

! Objective function to minimise :
                OBJ: SUM(I=1:DIM(2)) NCOMB(I) * L_COMB(I)  $
```

```
! Each small beam of each length must be produced :
  PRODL(K=1:DIM(1)):    NLONG(K) < SUM(I=1:DIM(2)) COMB(I,K) * NCOMB(I)

BOUNDS
  NCOMB(I=1:DIM(2)) .UI. NP

GENERATE
```

2. Seconde ébauche : uniformisation des poids lingots.

Dans sa seconde ébauche, le Programme Linéaire va prendre en considération la contrainte d'uniformisation des poids lingots (contrainte c). Les lingots ne sont en fait rien d'autre que les barres-mères sous forme de blocs d'acier. Nous pouvons donc à ce niveau d'abstraction identifier les lingots aux barres-mères, c'est-à-dire aux longueurs des combinaisons de découpe. Pour cette raison, nous dimensionnerons les lingots en mètres et non en tonnes. Affecter les barres-mères aux lingots revient maintenant à affecter chaque combinaison choisie à une longueur de lingot, de sorte qu'au plus trois (paramètre) longueurs de lingots différentes soient utilisées. l'ensemble des longueurs des lingots est l'ensemble des longueurs de toutes les combinaisons proposées.

Les données :

- Le couple traité :

  K : nombre de postes du couple. (N)

  k=1..K : indice sur les postes. (N)

  $Long_k$: longueur des poutrelles du $k^{ème}$ poste du couple. (R)

  $NLong_k$: nombre de poutrelles dans le $k^{ème}$ poste du couple. (N)

- Les combinaisons de longueur de poutrelles, établies par le module GENECOMB

  I : nombre de combinaisons établies. (N)

  i=1..I : indice sur les combinaisons. (N)

  $(Comb_{i,k})_{k=1..K}$: coefficients de la $i^{ème}$ combinaison, occurrences des $Long_k$ dans la $i^{ème}$ combinaison.

  (N**)

  $LComb_i$: longueur de la combinaison i :

  $LComb_i = |(Comb_{i,k})_{k=1..K}| = \Sigma_k\ Comb_{i,k} * Long_k$ (R)

- Le nombre maximum de poids lingots différents admis :

  Max_poids (N)

- Les poids lingots, établis par le module DOWEIGHT :

J :	nombre de poids lingots établis.	(N)
j=1..J :	indice sur les poids lingots.	(N)
L_PL_j :	longueur du $j^{ème}$ poids lingot, j=1..J	(R)

Les variables du Programme Linéaire :

- Les nombres d'utilisation des combinaisons de découpe de barres-mères :

$NComb_i$, i = 1..I : nombre de fois que la combinaison $(Comb_{i,k})_{k=1..K}$ est utilisée

- Les nombres d'utilisation des poids lingots :

N_PL_j, j = 1..J : nombre de fois que le poids lingot L_PL_j est utilisé

δ_j, j = 1..J : booléen valant 0 si N_PL_j=0, 1 si N_PL_j=1

Modélisation du Programme Linéaire :

fonction objective :

$$\text{minimiser} \left(\sum_{i=1}^{I} NComb_i * LComb_i \right)$$

tel que :

$$\forall k = 1..K : \sum_{i=1}^{I} NComb_i * Comb_{i,k} \geq NLong_k \quad \{ \text{ recouvrement du couple } \}$$

$$\forall i = 1..I : \sum_{i' : |Comb_{i'}| \geq |Comb_i|} NComb_{i'} \leq \sum_{j : L\_PL_j \geq |Comb_i|} N\_PL_j \quad \{ \text{ affectation des barres-mères dans des lingots au moins aussi grands} \}$$

$$\sum_{j=1}^{J} \delta_j \leq Max\_poids \quad \{ \text{ au plus Max\_poids poids différents } \}$$

$$\forall j = 1..J : \begin{cases} \delta_j \geq \dfrac{N\_PL_j}{1 + \sum_{k=1}^{K} NLong_k} \\ \delta_j \leq N\_PL_j \end{cases} \quad \{ \delta_j = \begin{cases} 0 \text{ si } N\_PL_j = 0 \\ 1 \text{ sinon} \end{cases} \}$$

Modules sous-jacents :

GENECOMB : inchangé.

DOWEIGHT : établit la liste des poids lingots :

Les poids lingots L_PL_j sont équivalents aux longueurs des combinaisons.

$$\{ L\_PLj \} = \{ \mid Comb_i \mid \} = \Sigma_k \, Comb_{i,k} * LComb_k$$

Les poids lingots sont arrondis à 50 kg près (interprété en mètres) et rendus uniques.

SOLUTION : inchangé.

Remarque importante :

La contrainte d'affectation des barres-mères nous assure qu'il y a au moins autant de lingots que de barres-mères. La minimisation de la quantité totale d'acier commandé, elle, nous assure qu'à l'optimum il y a exactement autant de lingots que de barres-mères.

Or cet optimum n'est que rarement atteint : pour des raisons de temps de calcul, le processus d'optimisation doit parfois être arrêté à la première solution entière trouvée. Il est donc important d'ajouter une contrainte vérifiant l'égalité du nombre de lingots et de barres-mères :

$$\sum_{i=1}^{I} NComb_i = \sum_{j=1}^{J} N\_PL_j$$

mais: Qu'en est-il de l'efficacité ?

Si effectivement le nombre de barres-mères et de lingots sont quelques fois différents lorsque seule la première solution entière est considérée, l'ajout de cette contrainte ne fournit pas toujours de gain dans la quantité d'acier totale commandée.

Ceci peut s'expliquer :

- les N_Pl_j seront grands car ils constituent une limite supérieure sur les contraintes d'affectation des barres-mères.
- l'ajout de la contrainte ci-dessus peut avoir pour conséquence une augmentation des $NComb_i$ plutôt qu'une diminution du nombre de poids lingots, car les $NComb_i$ sont elles limitées inférieurement par la contrainte du recouvrement du couple.

3. La gestion des sous-longueurs.

Nous allons maintenant gérer la perte statistique d'acier par rapport à la commande en tenant compte des contraintes d). Ceci revient à considérer en plus de l'ensemble de tous les poids lingots établis l'ensemble de ces poids lingots réduits du pourcentage statistique fixe de perte d'acier. Il faut s'assurer que les lingots commandés sont équitablement répartis dans chacun de ces ensembles des poids lingots.

Les données :

- Le couple traité :

K :	nombre de postes du couple.	(N)
k=1..K :	indice sur les postes.	(N)
$Long_k$:	longueur des poutrelles du $k^{ème}$ poste du couple.	(R)
$NLong_k$:	nombre de poutrelles dans le $k^{ème}$ poste du couple.	(N)

- Les combinaisons de longueur de poutrelles établies par le module GENECOMB:

I :	nombre de combinaisons établies.	(N)
i=1..I :	indice sur les combinaisons.	(N)
$(Comb_{i,k})_{k=1..K}$:	coefficients de la $i^{ème}$ combinaison, occurrences des $Long_k$ dans la $i^{ème}$ combinaison.	(N**)
$LComb_i$:	longueur de la combinaison i : $LComb_i = \|(Comb_{i,k})_{k=1..K}\| = \Sigma_k\ Comb_{i,k} * Long_k$	(R)

- Le nombre maximum de poids lingots différents admis :

Max_poids	(N)

- Le pourcentage statistique fixe de perte d'acier par rapport à la commande :

Pourcent	([0,1[ )

- Les poids lingots, établis par le module DOWEIGHT :

J :	nombre de poids lingots établis.	(N)
j=1..J :	indice sur les poids lingots.	(N)
$L1\_PL_j$:	longueur du $j^{ème}$ poids lingot, j=1..J	(R)
$L2\_PL_j$:	(1 - Pourcent) * longueur du $j^{ème}$ poids lingot, j=1..J	(R)

Les variables du Programme Linéaire :

- Les nombres d'utilisation des combinaisons de découpe de barres-mères :

  $NComb_i$, i = 1..I : nombre de fois que la combinaison $(Comb_{i,k})_{k=1..K}$ est utilisée.

- Les nombres d'utilisation des poids lingots :

  $N1\_PL_j$, j = 1..J : nombre de fois que le poids lingot L_PL1_j est utilisé.

  $N2\_PL_j$, j = 1..J : nombre de fois que le poids lingot L_PL2_j est utilisé.

  δ_j, j = 1..J : booléen valant 0 si N_PL1_j=0, 1 si N_PL1_j=1

Modélisation du Programme Linéaire :

fonction objective :

$$\text{minimiser}\left(\sum_{i=1}^{I} NComb_i * LComb_i\right)$$

tel que :

$$\forall k = 1..K: \sum_{i=1}^{I} NComb_i * Comb_{i,k} \geq NLong_k \quad \{ \text{ recouvrement du couple } \}$$

{ affectation des BM dans des lingots au moins aussi grands : }

$$\forall i = 1..I: \sum_{i':|Comb_{i'}|\geq|Comb_i|} NComb_{i'} \leq \sum_{j:\, L1\_PL_j \geq |Comb_i|} N1\_PL_j + \sum_{j:L2\_PL_j \geq |Comb_i|} N2\_PL_j$$

$$\sum_{j=1}^{J} \delta_j \leq Max\_poids \qquad \{ \text{ au plus Max\_poids poids différents } \}$$

$$\forall j = 1..J: \begin{cases} \delta_j \geq \dfrac{N1\_PL_j + N2\_PL_j}{1 + \sum_{k=1}^{K} NLong_k} \\ \delta_j \leq N1\_PL_j + N2\_PL_j \end{cases} \qquad \{ \delta_j = \begin{cases} 0 \text{ si } N\_PL_j = 0 \\ 1 \text{ sinon} \end{cases} \}$$

$$\forall j = 1..J: \begin{cases} N1\_Pl_j \leq N2\_Pl_j \\ N2\_Pl_j \leq N1\_Pl_j + 1 \end{cases} \qquad \{ \text{ perte statistique 1 barre-mère sur deux } \}$$

Modules sous-jacents :

GENECOMB : inchangé.

DOWEIGHT : inchangé.

SOLUTION : inchangé.

4. L'ajout du pourcentage statistique fixe d'acier.

Etant donné le pourcentage statistique de manque d'acier par rapport à la commande, une barre-mère sur deux, il est parfois nécessaire de commander une barre-mère augmentée afin de compenser la perte (contrainte e). Cela revient à proposer en plus des choix de poids lingots, l'ensemble de ces poids lingots augmentés du pourcentage statistique fixe de perte d'acier. Il suffit donc de modifier le module DOWEIGHT.

Les données :

Inchangées.

Les variables du Programme Linéaire :

Inchangées.

Modélisation du Programme Linéaire :

Inchangée.

Modules sous-jacents :

GENECOMB : inchangé.

DOWEIGHT :

On avait jusqu'à présent:

$$\{L\_Pl_j\} = \{L1\_Pl_j\} = \{\,|Comb_i|\,\}$$

maintenant, on redéfinit l'ensemble des poids lingots :

$$\{L\_Pl_j\}' = \{L\_Pl_j\} \cup \{(1+\alpha) * L\_Pl_j\}$$

avec α le pourcentage de compensation:

$$\alpha \equiv L\_Pl_j = (1\text{-Pourcent})\,[(1+\alpha)\,L\_Pl_j]$$

i.e. $\alpha = 1/(1\text{-Pourcent}) - 1$ avec $\alpha \in [0,\infty[$ et $Pourcent \in [0,1[$

SOLUTION : inchangé.

5. Les deux ébauches.

Pour prendre en compte cette contrainte (contrainte f), globale au couple complet, il suffit de considérer les barres-mères et les lingots 2 fois plus grands et d'exécuter le P.L., comme dans le cas d'une seule ébauche, sur base de ces données modifiées. Chaque lingot pourra ensuite être découpé en 2 selon les combinaisons de découpe choisies, qui seront scindées en 2 également. Ceci est possible car les longueurs de poutrelles sont restées inchangées, et il y en a en moyenne 2 fois plus par combinaison.

Les données :

Inchangées.

Les variables du Programme Linéaire :

Inchangées.

Modélisation du Programme Linéaire :

Inchangée.

Modules sous-jacents :

GENECOMB :

remplacer l'intervalle [Lg_min, Lg_max] des longueurs valables de barre-mère dans la génération des combinaisons par l'intervalle [2*Lg_min, 2*Lg_max].

DOWEIGHT : inchangé.

SOLUTION : inchangé.

6. Le respect de l'ordonnancement des postes.

L'ordonnancement des postes établi par les applications précédant la Commande d'Acier devrait idéalement être respecté (contrainte g). Il est clair que l'acceptation d'introduire un certain désordre au sein des postes contribue à l'obtention d'une meilleure optimisation de la Commande d'Acier considérée isolément. Le désordre introduit sera borné supérieurement par un paramètre : ce paramètre est la distance maximale acceptée entre deux numéros de postes présents au sein d'une même combinaison. Les combinaisons ne respectant pas cette distance maximale seront rejetées par la procédure ELIM du module GENECOMB. Une solution plus élégante et efficace est de ne générer les combinaisons de longueurs que pour les longueurs de poutrelles des postes apparaissant dans une fenêtre se déplaçant. Cette

solution est retenue et sera détaillée dans le chapitre suivant. La taille de la fenêtre sera la distance maximale admise entre deux postes au sein d'une barre-mère.

Les données :

Inchangées.

Les variables du Programme Linéaire :

Inchangées.

Modélisation du Programme Linéaire :

Inchangée.

Modules sous-jacents :

GENECOMB :
génération des combinaisons dans une fenêtre se déplaçant sur les postes du couple traité (voir chapitre suivant).

DOWEIGHT : inchangé.

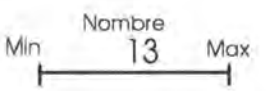
SOLUTION : inchangé.

7. La contrainte des lingotières.

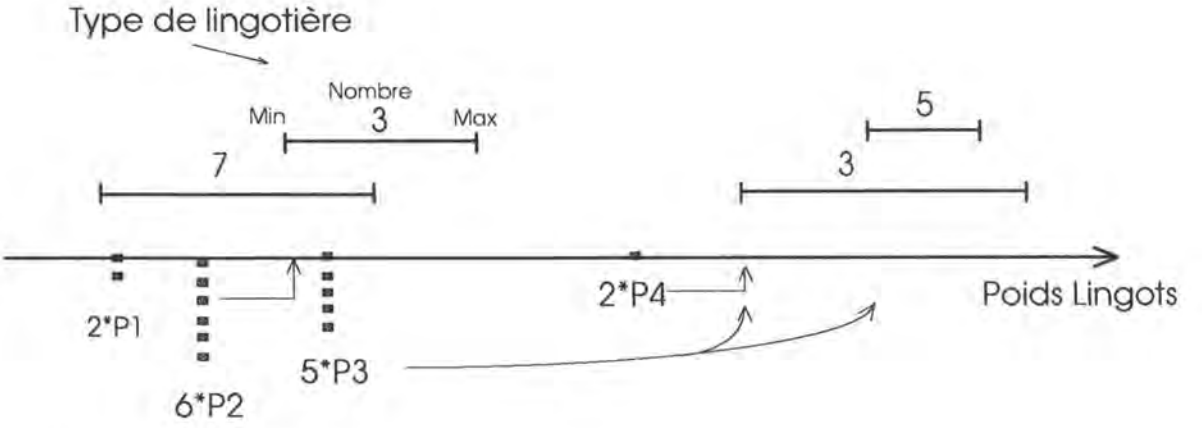
La contrainte des lingotières (contrainte h) est une contrainte d'affectation équivalente à l'affectation des barres-mères dans les lingots, quoique beaucoup plus délicate. Il s'agit ici de s'assurer que l'affectation aux lingotières disponibles des lingots commandés est possible.

Les lingotières sont classées en différents types caractérisés par des niveaux minimum et maximum de remplissage (représentés par [Min,Max]), correspondant aux poids minimum et maximum des lingots pouvant y être contenus. Le nombre de lingotières disponibles pour chacun des types varie.

Type de lingotière :



La contrainte peut être représentée schématiquement comme suit :



Contrainte du P.L. : on ne peut pas calculer effectivement l'affectation des lingots dans les lingotières (temps calcul!), mais il faut assurer qu'elle est possible.

Problème : on n'a pas de condition nécessaire et suffisante permettant de décider si tous les lingots sont affectables dans les lingotières [Min,Max] pouvant les contenir.

Nous allons donc assurer que l'affectation dans les lingotières est réalisable, éventuellement en plaçant des lingots dans des lingotières trop grandes (ce qui augmente les chutages). On considère pour cela que les lingotières ne sont plus caractérisées par les niveaux [Min,Max] de remplissage, mais par le niveau [Max] uniquement (c.-à-d. Min = 0). Certaines précautions doivent donc être prises, et un risque de grande perte d'optimisation subsiste.

Condition nécessaire et suffisante d'affectation dans des lingotières [Max] :

- similaire à la contrainte d'affectation des barres-mères dans les lingots.
- on assure qu'une affectation est toujours possible, éventuellement dans des lingotières de niveau Min trop élevé.
- affectation garantie mais risque d'une perte importante d'optimisation, car les lingots décalés dans des lingotières trop grandes le sont après optimisation.

Décaler les poids lingots ne se trouvant pas dans un intervalle [Min,Max] de type de lingotières vers le niveau Min immédiatement supérieur, préalablement à l'optimisation :

- prise en compte partielle du décalage des poids lingots.
- réduction des pertes post-optimisation.

Borner le nombre de lingots d'un poids donné par le nombre de lingotières pouvant le contenir :

☞ $\forall$ j=1..J : N_PLj $\leq$ #{ L | L est lingotière $\wedge$ L $\supset P_j$ }

☞ réduction des pertes post-optimisation.

Le risque de perte importante d'optimisation dû à des décalages de lingots dans une lingotière de niveau minimum trop grand subsiste. Néanmoins, si le nombre de lingotières de chaque type est suffisamment élevé, ce risque reste faible.

Les données :

- Le couple traité :

K :	nombre de postes du couple.	(N)
k=1..K :	indice sur les postes.	(N)
$Long_k$:	longueur des poutrelles du $k^{ème}$ poste du couple.	(R)
$NLong_k$:	nombre de poutrelles dans le $k^{ème}$ poste du couple.	(N)

- Les combinaisons de longueur de poutrelles, établies par le module GENECOMB :

I :	nombre de combinaisons établies.	(N)
i=1..I :	indice sur les combinaisons.	(N)
$(Comb_{i,k})_{k=1..K}$:	coefficients de la $i^{ème}$ combinaison, occurrences des $Long_k$ dans la $i^{ème}$ combinaison.	(N**)
$LComb_i$:	longueur de la combinaison i : $LComb_i = \lvert(Comb_{i,k})_{k=1..K}\rvert = \Sigma_k\ Comb_{i,k} * Long_k$	(R)

- Le nombre maximum de poids lingots différents admis :

Max_poids	(N)

- Le pourcentage statistique fixe de perte d'acier par rapport à la commande :

Pourcent	([0,1[ )

- Les poids lingots, établis par le module DOWEIGHT :

J :	nombre de poids lingots établis.	(N)
j=1..J :	indice sur les poids lingots.	(N)
$L1\_PL_j$:	longueur du $j^{ème}$ poids lingot, j=1..J	(R)
$L2\_PL_j$:	(1 - Pourcent) * longueur du $j^{ème}$ poids lingot, j=1..J	(R)

- Les types de lingotières :

L :	nombre de types de lingotières.	(N)
l=1..L :	indice sur les lingotières.	(N)

N_lingot_l : nombre de lingotières de type l, l=1..L (R)

Min_lingot_l : niveau minimum de remplissage des lingotières de type l, exprimé en mètres. (R)

Max_lingot_l : niveau maximum de remplissage des lingotières de type l, exprimé en mètres. (R)

Les variables du Programme Linéaire :

- Les nombres d'utilisation des combinaisons de découpe de barres-mères :

  $NComb_i$, i = 1..I : nombre de fois que la combinaison $(Comb_{i,k})_{k=1..K}$ est utilisée.

- Les nombres d'utilisation des poids lingots :

  $N1\_PL_j$, j = 1..J : nombre de fois que le poids lingot L_PL1_j est utilisé.

  $N2\_PL_j$, j = 1..J : nombre de fois que le poids lingot L_PL2_j est utilisé.

  δ_j, j = 1..J : booléen valant 0 si N_PL1_j=0, 1 si N_PL1_j=1

- Les nombres d'utilisation des lingotières :

  $Lingot_l$, l = 1..L : nombre de lingotières de type l utilisées.

Modélisation du Programme Linéaire :

fonction objective :

$$\text{minimiser}\left(\sum_{i=1}^{I} NComb_i * LComb_i\right)$$

tel que :

$$\forall k = 1..K: \sum_{i=1}^{I} NComb_i * Comb_{i,k} \geq NLong_k \quad \{ \text{ recouvrement du couple } \}$$

{ affectation des BM dans des lingots au moins aussi grands : }

$$\forall i = 1..I: \sum_{i':|Comb_{i'}| \geq |Comb_i|} NComb_{i'} \leq \sum_{j:L1\_PL_j \geq |Comb_i|} N1\_PL_j + \sum_{j:L2\_PL_j \geq |Comb_i|} N2\_PL_j$$

$$\sum_{j=1}^{J} \delta_j \leq Max\_poids \quad \{ \text{ au plus Max\_poids poids différents } \}$$

$$\forall j = 1..J: \begin{cases} \delta_j \geq \dfrac{N1\_PL_j + N2\_PL_j}{1 + \sum_{k=1}^{K} NLong_k} \\ \delta_j \leq N1\_PL_j + N2\_PL_j \end{cases} \quad \{ \delta_j = \begin{cases} 0 \text{ si } N\_PL_j = 0 \\ 1 \text{ sinon} \end{cases} \}$$

$$\forall j = 1..J: \begin{cases} N1\_Pl_j \le N2\_Pl_j \\ N2\_Pl_j \le N1\_Pl_j + 1 \end{cases}$$ { perte statistique 1 barre-mère sur deux }

{ affectation des lingots dans les lingotières : }

$$\forall j = 1..J: \sum_{j':L1\_PL_{j'} \ge L1\_PL_j} (N1\_PL_{j'} + N2\_PL_{j'}) \le \sum_{\substack{l=1..L \\ Max\_lingot_l \ge L1\_PL_j}} Lingot_l$$

{ diminuer les décalages de lingots vers des lingotières plus grandes : }

$$\forall j = 1..J: N1\_PL_j + N2\_PL_j \le \sum_{\substack{l=1..L \\ Min\_lingot_l \ge L1\_PL_j \\ Max\_lingot_l \le L1\_PL_j}} N\_Lingot_l$$

Modules sous-jacents :

GENECOMB : inchangé.

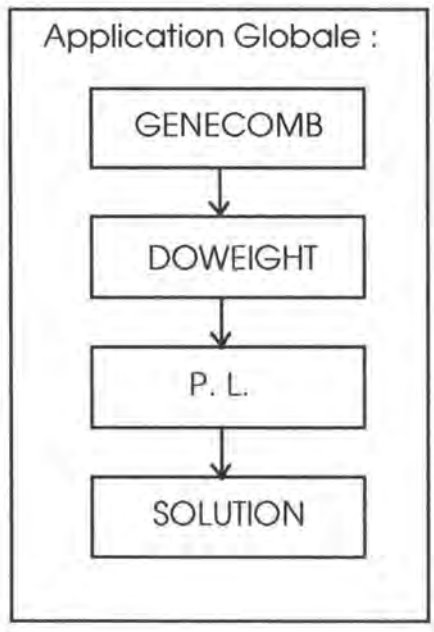
DOWEIGHT :

Décaler les poids lingots ne se trouvant pas dans un intervalle [Min,Max] de type de lingotières vers le niveau Min immédiatement supérieur préalablement.

SOLUTION : inchangé.

8. Modélisation complète du Programme Linéaire.

Reprenons à des fins de synthèse la modélisation complète du P.L., c.-à-d. celle établie à la fin de l'étape précédente. Les modules sous-jacents au P.L. seront eux spécifiés dans le chapitre suivant.



Modélisation avec le modélisateur XPRESS-MP de Dash Associates :

```
!                ==========
!                | Steel Order |
!                ==========

MODEL  CA_11          ! MODELISATION 1.1

TABLES
  DIM(5)              ! DIM(1) = Number of small beams different lengths (K).
                      ! DIM(2) = Number of valid combinaisons (I).
                      ! DIM(3) = Number of valid ingot lengths (J).
                      ! DIM(4) = Maximum number of different ingot lengths that can be used.
                      ! DIM(5) = Number of ingot cauldron types (L).

DISKDATA
  DIM(1) = DIM.XPR

TABLES
  LONG(DIM(1))            ! LONG(K) = K-th small beam length.
  NLONG(DIM(1))           ! NLONG(K) = number of small beam of K-th length to produce.
  COMB(DIM(1),DIM(2))     ! COMB(I,K) = number of small beams of  K-th length in the
                          ! I-th combinaison.
  L_COMB(DIM(2))          ! L_COMB(I) = Length of MB expected if made from I-th combinaison.
  L1_PL(DIM(3))           ! L1_PL(J) = 100% of J-th ingot length.
  L2_PL(DIM(3))           ! L2_PL(J) = 97% of J-th ingot length.
  N_INGOT((DIM5))         ! N_INGOT(L) = number of ingot cauldron of type L.
  MIN_INGOT(DIM(5))       ! MIN_INGOT(L) = Minimum level of ingot cauldron of type L.
  MAX_INGOT(DIM(5))       ! MAX_INGOT(L) = Maximum level of ingot cauldron of type L.

DISKDATA
  LONG(1) = LONG.XPR
  NLONG(1) = NLONG.XPR
  COMB(1,1) = COMB.XPR
  L1_PL(1) = POIDS.XPR
  MIN_INGOT(1) = MIN_LING.XPR
  MAX_INGOT(1) = MAX_ING.XPR
  N_INGOT(1) = N_INGOT.XPR

VARIABLES
  NCOMB(DIM(2))           ! NCOMB(I) = how many times combinaison I is used.
  N1_PL(DIM(3))           ! N1_PL(J) = how many times L1_PL(J) is used.
  N2_PL(DIM(3))           ! N2_PL(J) = how many times L2_PL(J) is used.
  DELTA(DIM(3))           ! DELTA(J) = binary variables.
                          !       value is
                          !           0: J-th ingot length IS NOT used (N1_PL = N2_PL = 0)
                          !           1: J-th ingot length IS used (N1_PL + N2_PL <> 0)
  INGOT(DIM(5))           ! INGOT(L) = number of ingot cauldron of type L used.

ASSIGN
  L_COMB(I=1:DIM(2)) = SUM(K=1:DIM(5)) LONG(K+SHIFT(I))*COMB(I,K)
  L2_PL(J=1:DIM(3)) = .97*L1_PL(J)
  NP=SUM(K=1:DIM(1)) NLONG(K)      ! Small beams total number to produce.

CONSTRAINTS
```

```
! Objective function to minimise :
      OBJ: SUM(J=1:DIM(3)) L1_PL(J)*N1_PL(J) + SUM(J=1:DIM(3)) L1_PL(J)*N2_PL(J) $

! Each small beam of each length must be produced :
      PRODL(K=1:DIM(1)):  NLONG(K) < SUM(I=1:DIM(2)) COMB(I,K) * NCOMB(I)

! Affectation problem : it's possible to produce all combinaisons used
! from chosen ingot lengthes ( proved formula )
      AFF(I=1:DIM(2)): SUM(I1=1:DIM(2)|L_COMB(I1) .GE. L_COMB(I)) NCOMB(I1)<&
                  SUM(J=1:DIM(3)|L1_PL(J) .GE. L_COMB(I)) N1_PL(J) &
               + SUM(J=1:DIM(3)|L2_PL(J) .GE. L_COMB(I)) N2_PL(J)

! No more than DIM(4) different ingot lengthes :
   MAX_P: SUM(J=1:DIM(3)) DELTA(J) < DIM(4)
   NB_P1(J=1:DIM(3)): DELTA(J) < N1_PL(J) + N2_PL(J)
   NB_P2(J=1:DIM(3)): DELTA(J) > 1./NP*N1_PL(J) + 1./NP*N2_PL(J)

! There are as many L1_PL ingot lengthes as L2_PL ingot lengthes.
! ( N1_PL = N2_PL  ) or  ( N2_PL = N1_PL + 1 ) :
   PROP1(J=1:DIM(3)): N2_PL(J) - N1_PL(J) < 1
   PROP2(J=1:DIM(3)): N2_PL(J) - N1_PL(J) > 0

! Affectation problem : it's possible to produce all ingots used from ingot cauldrons.
      AFING(J=1:DIM(3)): SUM(J1=1:DIM(3)|L1_PL(J1) .GE. L1_PL(J)) N1_PL(J1) + &
                         SUM(J1=1:DIM(3)|L1_PL(J1) .GE. L1_PL(J)) N2_PL(J1) < &
                         SUM(L=1:DIM(5)|MAX_INGOT(L) .GE. L1_PL(J)) INGOT(L)
      MOVE(J=1:DIM(3)): N1_PL(J) + N2_PL(J) < &
                         SUM(L=1:DIM(5)|MIN_INGOT(L) .GE. L1_PL(J) .AND. &
                                        MAX_INGOT(L) .LE. L1_PL(J))  NINGOT(L)

BOUNDS
  NCOMB(I=1:DIM(2)) .UI. NP
  N1_PL(J=1:DIM(3)) .UI. NP
  N2_PL(J=1:DIM(3)) .UI. NP
  DELTA(J=1:DIM(3)) .BV.
  INGOT(L=1:DIM(5)) .UI. NINGOT(L)

GENERATE

!     Parameters used :              Directives used :
!     ----------------               ----------------
!
!       MXANOD = 50000                   PR   DELTA???   1
!       NGEMAX = 1000
!       INTMAX = 5000
!       NDSEL2 = 2
```

IV. Les modules sous-jacents au P.L.

A. Générer des combinaisons de poutrelles des postes : les modules GENECOMB, STOCOMB, ELIM, EVAL.

1. Générer des combinaisons : le module GENECOMB.

Le module GENECOMB permet de générer toutes les combinaisons de longueur de poutrelles des postes du couple (dérivé,nuance) traité, de façon à former des barres-mères susceptibles d'être commandées à l'aciérie.

Une combinaison est représentée par des coefficients, chaque coefficient indiquant l'occurrence d'une longueur de poutrelle au sein de la barre-mère:

$$Comb_i \equiv (Comb_{i,k})_k : \sum_k (Comb_{i,k} * Long_k) = Barre\text{-}Mère$$

Notation: $Comb_i = Comb_{i,k}$

$$|Comb_i| = \sum_k (Comb_{i,k} * Long_k)$$

Exemple :

Soit la $i^{ème}$ combinaison proposée : $2 \times 15\ m + 4 \times 6\ m + 3 \times 10\ m = 84\ m$

Nous avons :

$(Comb_{i,k})_k = (2,4,3)$

$LComb_i = 84$

Critères de la génération:

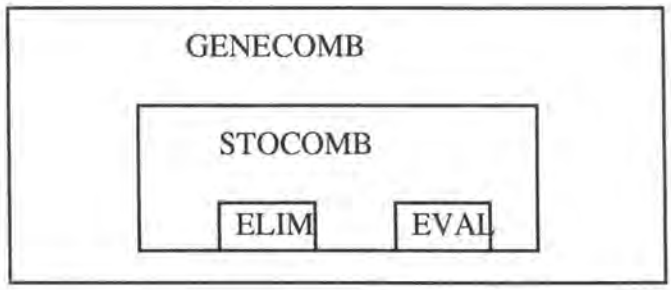
- Les combinaisons générées sont l'ensemble exhaustif de toutes les combinaisons des longueurs de poutrelles telles que la longueur de combinaison soit comprise dans un intervalle [Lmin_BM, Lmax_BM] et telles que l'occurrence maximum d'une longueur de poutrelle au sein des combinaisons est spécifiée.

$$Comb_i \text{ générée} \Leftrightarrow \begin{cases} |Comb_i| \in [L\min\_BM ; L\max\_BM] \\ \forall k : Comb_{i,k} \leq \max_k \end{cases}$$

- Les combinaisons générées sont validées, pondérées et triées selon l'ordre croissant des pondérations.

☞ Si plusieurs combinaisons ont la même pondération, le tri s'effectue selon l'ordre de génération, c'est-à-dire l'ordre lexicographique décroissant des k-uplets de coefficients de combinaisons.

le module GENECOMB fait appel à plusieurs procédures :



Notion de fenêtre sur les longueurs du couple:

La durée de génération des combinaisons de longueurs d'un couple varie de façon non linéaire avec la taille du couple.

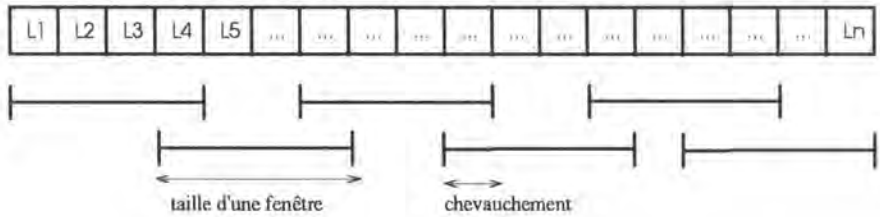
Il résulte que le module GENECOMB ne peut pas travailler sur des couples de grande taille en des temps restreints. D'autre part, dès que le nombre de longueurs à traiter est élevé, le nombre de coefficients nuls dans les combinaisons devient important.

Le module GENECOMB ne va donc pas être appelé pour générer des combinaisons relatives à toutes les longueurs du couple, mais pour générer les combinaisons relatives aux longueurs apparaissant dans une fenêtre qui va se déplacer tout le long du couple.

Le module GENECOMB sera appelé lors de chaque déplacement d'une fenêtre sur les longueurs de poutrelles des postes du couple traité.

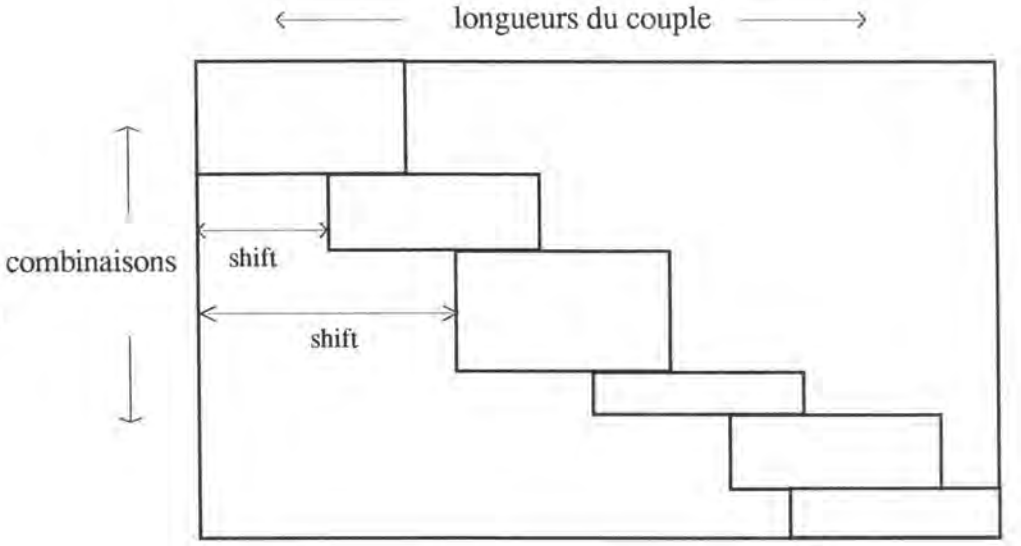
Schématiquement:

tableau des longueurs du couple:



fenêtre de taille constante, avec ou sans chevauchement.

La taille de la fenêtre et la longueur du chevauchement étant constants, on se retrouve avec une matrice des combinaisons diagonale par blocs. Le nombre de combinaisons générées par fenêtre est variable. Pour économiser la mémoire, on ne gardera que les blocs car il est inutile de remplir plus de la moitié d'un tableau avec des zéros !



Conséquences immédiates:

Avant: $Comb_i \equiv (Comb_i)_k$, k = 1..taille du couple.

$|Comb_i| = \Sigma_k Comb_{i,k} * Long_k$

Maintenant: $Comb_i = ((Comb_i)_k, shift)$, k = 1..taille fenêtre

et $|Comb_i| = \Sigma_k Comb_{i,k} * Long_{k+shift}$

Shift = décalage de la fenêtre (n° du poste précédant la fenêtre)

Intérêt du fenêtrage:

- les longueurs n'apparaissant pas dans une même fenêtre, n'apparaissent pas dans une même combinaison : donc cela restreint à la taille de la fenêtre le désordre dans les longueurs de poutrelles des postes du couple traité au sein d'une barre-mère.
- les combinaisons ne sont composées que d'un faible nombre de coefficients : d'où gain important de mémoire, et dans une grande combinaison peu de coefficients sont non nuls (car $|comb| \geq Lmin\_BM$)
- la génération des combinaisons est beaucoup plus rapide : x générations de combinaisons de y longueurs sont au total plus rapides que 1 génération de x*y longueurs.

☞ la génération est rapide même pour de très grands couples : la variation de temps de génération est maintenant linéaire par rapport au temps de génération sur une fenêtre.

Nombre de déplacements de la fenêtre:

Soient N : taille du couple (nombre de postes)

n : taille fenêtre

c : chevauchement

E[x] : partie entière de x

alors Nombre de déplacements fenêtre $= E\left[\frac{N}{n-c}\right]+1$

Nombre de combinaisons au maximum par fenêtre :

$$E\left[\frac{nbre\_total\_comb}{E\left[\frac{N}{n-c}\right]+1}\right]$$

2. Mémoriser les combinaisons générées: le module STOCOMB.

Le module STOCOMB se charge de la gestion des combinaisons générées par GENECOMB. Il mémorise les combinaisons reçues de GENECOMB après validation par appel au module ELIM et pondération par appel au module EVAL.

Les combinaisons sont mémorisées dans un tableau dont la taille est limitée par un nombre maximum de combinaisons pouvant être retenues.

Le tableau des combinaisons retenues est trié selon l'ordre croissant des pondérations. A tout instant, il correspond aux meilleures combinaisons parmi toutes celles qui ont été générées par GENECOMB.

3. Valider une combinaison : le module ELIM.

Le module ELIM (sous-module de GENECOMB) reçoit une combinaison et renvoie un booléen d'acceptation ou de rejet de la combinaison selon des critères d'élimination.

Une combinaison est valide si elle n'est pas éliminée.

Une combinaison est éliminée si elle satisfait au moins un critère d'élimination.

Les critères d'élimination (validation) des combinaisons:

Les critères d'élimination sont ceux précisés dans les contraintes de la Commande d'Acier (contraintes a et b). Ils ne sont pas plus explicités car triviaux dans leur implémentation.

4. Pondérer une combinaison : le module EVAL.

Le module EVAL (sous-module de GENECOMB) a pour rôle d'évaluer une combinaison. Il reçoit une combinaison non rejetée et renvoie un coefficient de pondération pour cette combinaison selon des critères d'évaluation.

Les critères de pondération des combinaisons:

La pondération d'une combinaison est sa punition: plus une combinaison est intéressante, plus sa pondération sera proche de 0.

La pondération d'une combinaison au sens de tous les critères est la somme des pondérations de chacun des critères.

Un seul critère de pondération a été retenu. Celui-ci s'avère très efficace pour ne retenir que les combinaisons intéressantes au sens du couple donné.

critère 1 (unique) :

Soient :

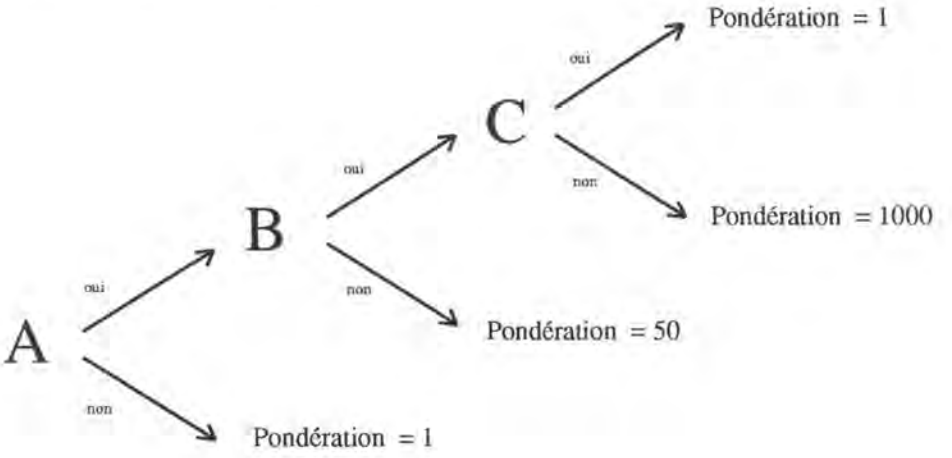
c : la combinaison à pondérer,

A : la proposition "toutes les longueurs de poutrelles référencées par les coefficients de la combinaison c sont déjà présentes dans au moins une autre combinaison",

B : la proposition "tous les coefficients de la combinaison c sont non nuls",

C : la proposition "le rapport c1/c2, c1≥c2, de deux quelconques coefficients consécutifs de la combinaison c est de l'ordre du rapport des occurrences, dans l'extrait, des longueurs de poutrelles concernées par les coefficients. Deux nombres sont considérés de même ordre s'ils ont même partie entière."

Le critère 1 s'énonce alors :



Le noyau du critère consiste en la proposition C. Elle ne peut cependant être validée que moyennant la vérification préalable des propositions A et B.

B. Proposer des poids lingots: le module DOWEIGHT.

Le module DOWEIGHT sert à établir une liste des poids lingots valides à partir desquels des barres-mères pourront être produites. Les poids calculés seront proposés au P.L.

Algorithme:

1. les lingots sont équivalents aux barres-mères :

$$\{L\_PL_j\} = \{\,|Comb_i|\,\}$$

2. la contrainte du pourcentage de perte par rapport à la commande :

$$\{L\_PL_j\}' = \{L\_PL_j\} \cup \{(1+\alpha)L\_PL_j\}$$

avec α le pourcentage de compensation $= \dfrac{1}{1 - pct\_fixe\_perte} - 1$

$$Pourcent\_fixe\_perte \in [0,1[$$

3. la contrainte des lingotières :

   Remplacer dans $\{Cple\_Wght_j\}'$ tous les $Cple\_Wght_j$ ne se trouvant pas dans un intervalle $[Min\_Lingot_l, Max\_Lingot_l]$ de type de lingotières par le niveau minimum du type de lingotière immédiatement supérieur.

   si aucun type de lingotière ne peut contenir $Cple\_Wght_j$, le supprimer.

4. les poids lingots sont arrondis à 50 kg près :

5. les poids lingots sont rendus uniques.

C. Interpréter la solution du P.L. : le module SOLUTION.

La solution calculée par le P.L. est donnée en termes de valeurs numériques pour chaque variable traitée, dans un fichier sous un format propre au logiciel.

Le module SOLUTION extrait toutes ces valeurs en lisant le fichier et en réécrivant les informations de façon compréhensible.

Les opérations effectuées sont:

- extraction du nombre de fois que chaque combinaison de découpe est utilisée du fichier des résultats du P.L.
- extraction du nombre de fois que chaque poids lingots proposé est utilisé du fichier des résultats du P.L.
- détermination de la combinaison de découpe de chaque lingot.
- affectation des lingots dans les lingotières.

Extraction des combinaisons de découpes (barres-mères) :

Chaque $Comb_i$ telle que $NComb_i \neq 0$ représente la découpe d'une barre-mère devant être commandée $NComb_i$ fois.

Extraction des poids lingots :

Normalement, la somme des nombres d'utilisation des combinaisons de découpes est égal à la somme des nombres d'utilisation des poids lingots. Cette égalité n'est cependant atteinte qu'à l'optimum et donc rarement vérifiée pour des raisons de temps calcul (voir description de la contrainte, chapitre III.B.2). Les seuls lingots effectivement commandés seront ceux pour lesquels une barre-mère (combinaison de découpe) a été affectée, selon l'algorithme donné ci-après. L'éventuel surplus de lingots doit être ignoré, rendant ainsi la contrainte supplémentaire proposée et discutée (chapitre III.B.2) inutile.

Algorithme d'affectation des combinaisons de découpe (barres-mères) aux lingots :

- Soit C la combinaison la plus petite non encore affectée à un lingot.
- le lingot découpé par cette combinaison C est le plus petit lingot de longueur supérieure à |C|.

en effet:

algorithme équivalent à la formule de vérification de possibilité d'affectation.

Algorithme d'affectation des lingots aux lingotières :

- soit P le plus grand de tous les lingots non encore affectés.

- la lingotière affectée à P est donnée par:

    - soit L l'ensemble des lingotières non encore affectées à un lingot, pouvant contenir P (i.e. tq Min_lingot ≤ P ≤ Max_lingot)

    - si L≠∅,

        soit l la lingotière ∈ L tq l ait le PLUS GRAND niveau MINIMUM de remplissage.

    sinon,

        soit l la lingotière ∈ L tq l ait le PLUS PETIT niveau MINIMUM de remplissage immédiatement supérieur à P

    - l est la lingotière affectée à P

en effet:

Puisque le prochain lingot ne peut être plus grand que celui traité, cette méthode d'affectation nous assure de choisir la moins gourmande des lingotières valides, au sens de sa capacité à contenir les prochains lingots. Remarquons que les résultats du P.L. en termes de lingotières utilisées sont ignorés. Leur seule utilité est au sein du P.L. d'assurer la possibilité d'affectation des lingots aux lingotières.

V. La gestion des grands couples

A. Le partitionnement du couple.

1. Nécessité.

Les supports logiciel et matériel sont à l'origine de la limite sur la taille des problèmes traitables, et ce, quelle que soit la modélisation.

En ce qui concerne notre Programme Linéaire, cette limite se traduit par le fait qu'il ne peut accepter qu'un nombre limité de combinaisons de découpe proposées, indépendamment de la taille du couple. Il en résulte que les résultats se dégradent fortement avec l'augmentation de la taille des couples traités.

En outre, il est plus intéressant de produire des résultats cumulés sur plusieurs partitions d'un couple plutôt que sur le couple entier. Les résultats cumulés sont meilleurs et plus rapides.

- ☞ Plus rapides: car moins de longueurs donc moins de contraintes
- ☞ Meilleurs: car au total 2 fois plus de combinaisons sont proposées, et l'approximation de la solution optimale sera plus fine sur une même durée totale puisque chaque approximation est atteinte plus rapidement.

C'est pourquoi le partitionnement des grands couples est indispensable. C'est l'application globale qui va prendre en charge la gestion de ce partitionnement. Dans le cas d'un partitionnement du couple traité, il faut exécuter le P.L. sur chacune des partitions. Les données envoyées au P.L. doivent donc correspondre à une partition et les résultats doivent être considérés en conséquence.

REMARQUE :

L'emploi des mots "partitionnement" et "partition" n'est pas entièrement correct, car cela sous-entend qu'aucun chevauchement des partitions n'est admis.

Or ce n'est pas le cas. Afin d'augmenter la transparence de la découpe d'un couple en plusieurs parties, et afin de n'influencer qu'au minimum les résultats, il est préférable de considérer un certain chevauchement des parties du couple.

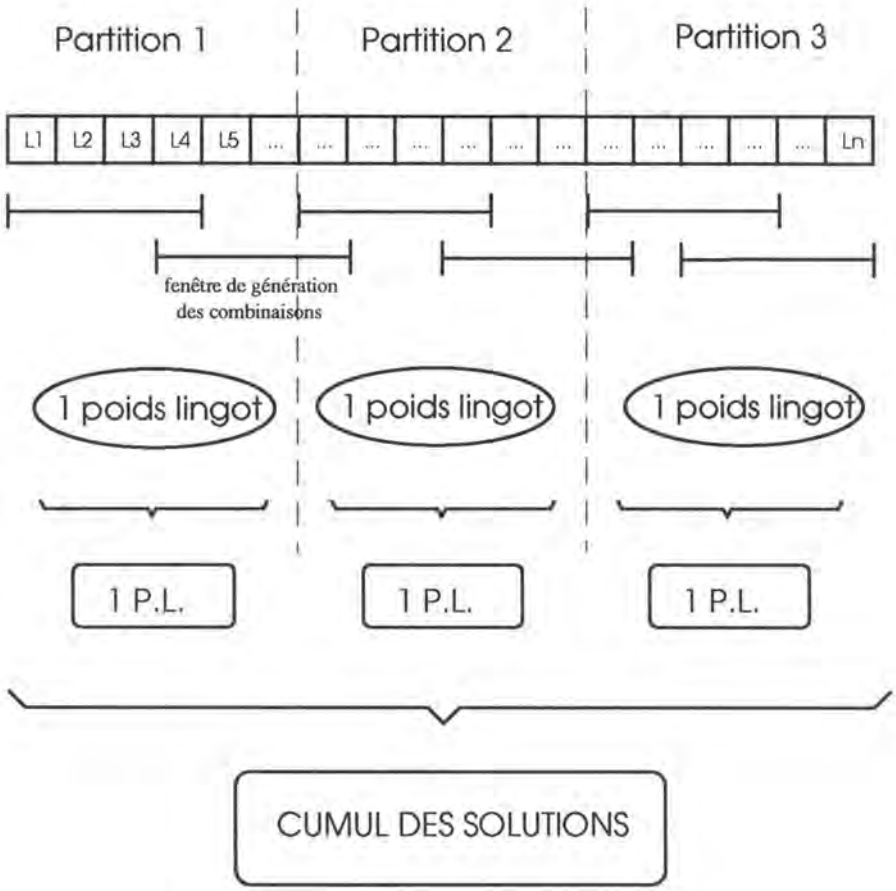
Les mots "partitionnement" et "partition" seront cependant utilisés, par abus de langage.

2. Possibilités de partitionnement et problèmes.

Avant de recourir au partitionnement du couple, il faut vérifier que traiter un couple en plusieurs parties ne nie aucune contrainte.

En traitant le couple par partitions, nous allons exécuter le P.L. sur chaque partition et additionner les solutions de chaque P.L. pour obtenir une solution pour le couple entier. Un problème apparaît tout de suite: c'est le respect de la contrainte portant sur le nombre maximum de poids lingots admis, et le respect de la contrainte des lingotières. Ce sont les seules contraintes qui ont une incidence sur le couple complet (problème d'affectation, global au couple). En effet, si elles sont vérifiées pour chaque partition, l'addition des résultats peut ne pas donner de résultat global valable. Il ne faut pas pour autant jeter l'idée d'un partitionnement du couple traité, car nous la savons indispensable. Envisageons donc une solution.

Solution 1:



pour chaque partition,

* diviser le nombre maximum de poids lingots admis par le nombre de partitions,
* biffer du stock des lingotières celles affectées dans les précédentes partitions,

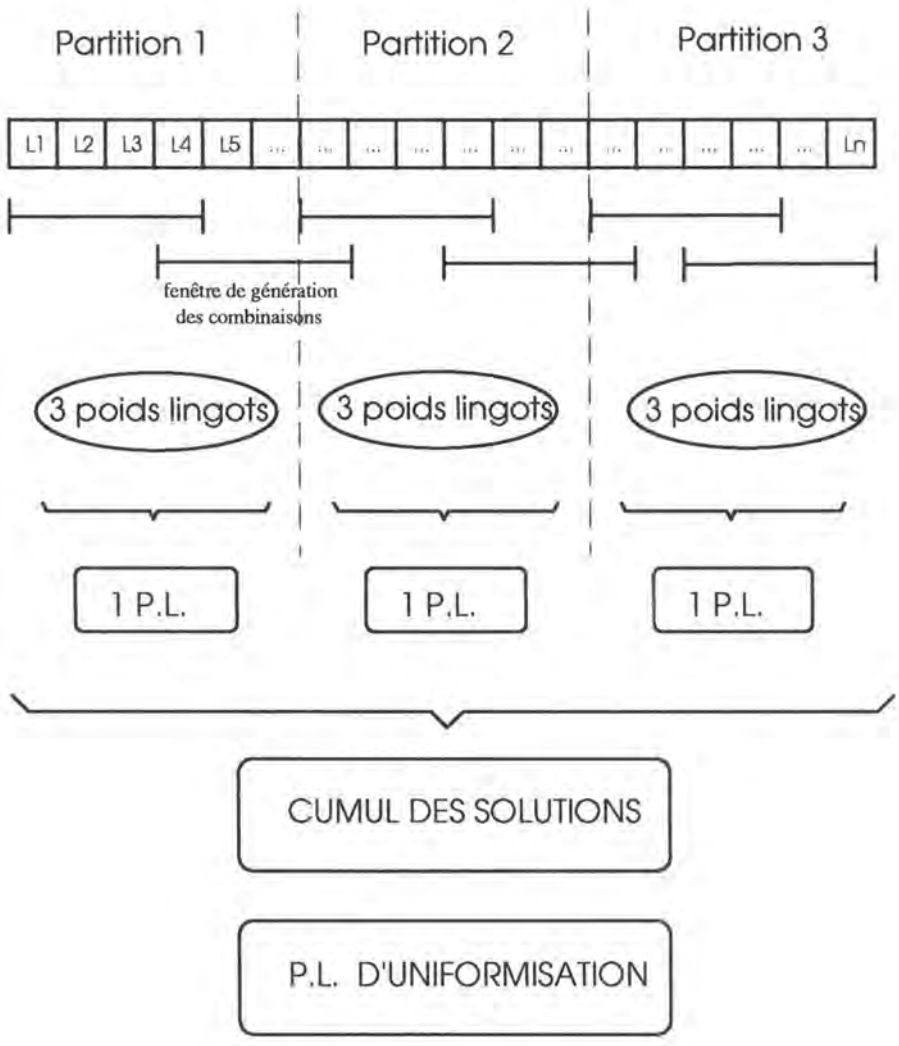
* cumul des solutions.

inconvénients :

- perte importante sur l'optimisation car optimisations locales sur un nombre trop faible de poids lingots admis.
- le stock de lingotières des dernières partitions est désavantageusement conditionné par le choix des lingotières utilisées dans les partitions précédentes.
- le nombre de partitions est limité au nombre maximum de poids lingots admis (généralement : 3), donc taille maximum de couple traitable très limitée.

conclusion: solution refusée.

Solution 2:



pour chaque partition,

* après exécution du P.L., on ne retient que les combinaisons choisies (pas leurs nombres d'utilisations respectifs) et les lingotières affectées, les autres résultats sont ignorés.

* biffer du stock des lingotières celles affectées dans chaque partition.

Lorsque le P.L. a été exécuté sur chaque partition,

* réexécuter le P.L. globalement au couple complet, avec pour combinaisons proposées, les combinaisons choisies par le P.L. sur chaque partition. On rappelle donc DOWEIGHT et on recalcule le nombre de fois que l'on utilise les barres-mères (nombre d'utilisation de chaque combinaison).

Avantages :

- l'optimisation reste globale au couple complet, sur base des combinaisons qui sont les meilleures au sens de tous les critères et au sens du couple, puisque résultant des optimisations précédentes.
- facile à mettre en oeuvre.

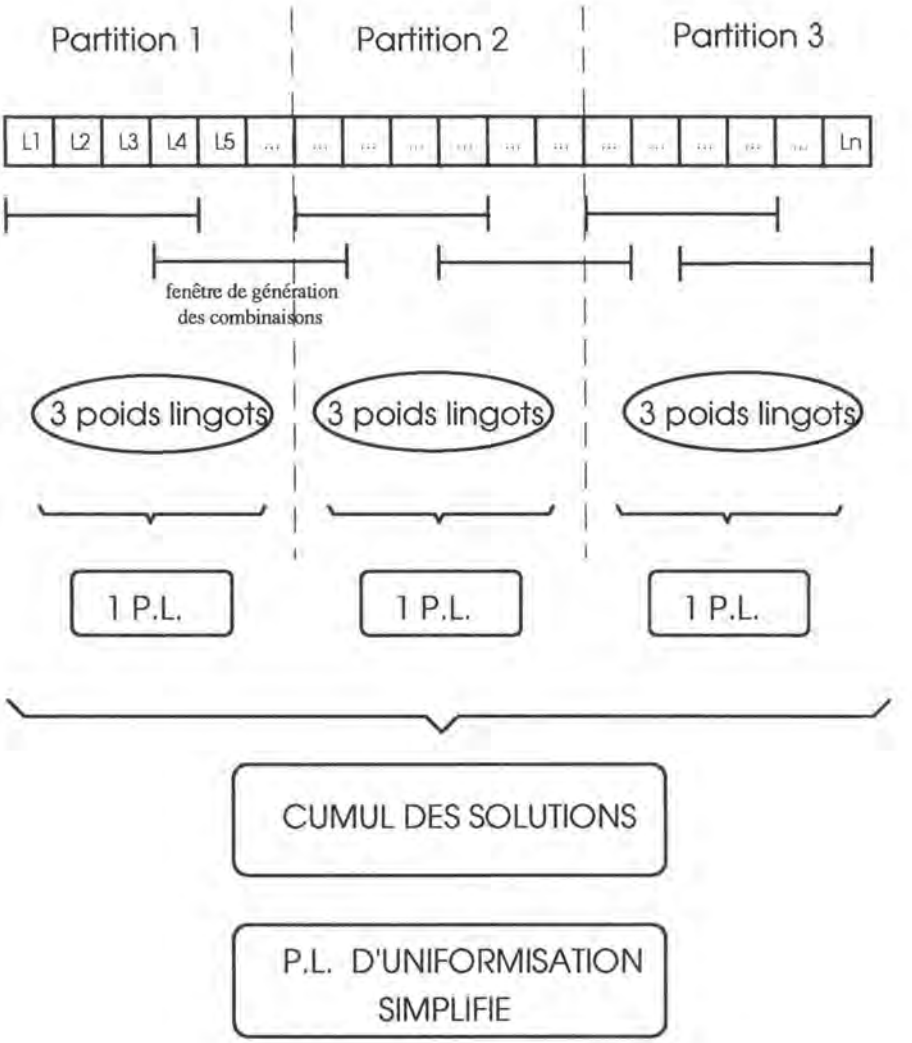
Inconvénients :

- le nombre de combinaisons retenues par les P.L. sur chaque partition n'est pas négligeable. Il y a donc une limite sur le nombre de partitions, c'est-à-dire sur la taille maximale d'un couple traitable.
- un risque subsiste concernant la contrainte des lingotières, voir ci-après : "la contrainte des lingotières mise en défaut"
- exécuter le P.L. une fois de plus que le nombre partitions.

Conclusion:

- la taille maximale d'un couple traitable se situe très largement au-delà de celles des problèmes réels à traiter.
- le risque concernant la contrainte des lingotières peut être inexistant, voir ci-après : "la contrainte des lingotières mise en défaut"
- cette solution est très intéressante quant aux résultats de la solution finale, car optimisation globale au couple.
- solution retenue.

Solution 3:



pour chaque partition,

* biffer du stock des lingotières celles affectées dans les précédentes partitions.

* après exécution du P.L., retenir les combinaisons de découpe et leurs quantités respectives utilisées, ainsi que les lingotières utilisées. Les lingots utilisés sont ignorés.

après exécution du P.L. pour chaque partition.

* trouver la meilleure affectation des barres-mères retenues sur chaque partition dans des lingots, et la meilleure affectation de ces lingots dans les lingotières (i.e. réexécuter le P.L. SANS CHOIX DE COMBINAISONS ET SANS RECALCUL DU NOMBRE DE FOIS QUE L'ON UTILISE UNE BARRE-MÈRE).

Avantages :

- plus de limite sur la taille maximale d'un couple traitable, car les combinaisons ne sont plus calculées (choisies) dans le P.L. global au couple complet.
- exécution plus rapide car le P.L. d'uniformisation est très réduit.

Inconvénients :

- la contrainte des lingotières n'est plus vérifiée! En effet: elle l'est pour chaque partition, avec nombre maximal de poids lingots admis ; globalement, elle est vérifiée avec un nombre de poids lingots largement supérieur au maximum admis. Réduire le nombre de poids lingots sans recalculer les lingots, c'est augmenter les lingots d'un certain poids au poids des lingots immédiatement supérieurs. Plus de lingotières sont donc nécessaires pour couler les lingots supplémentaires de ce poids. Il n'est pas certain que des lingotières de ce type sont encore disponibles.

  $\Rightarrow$ non certain.

Conclusion: Solution rejetée.

3. La contraintes des lingotières mise en défaut.

La possibilité d'affectation des lingots dans les lingotières, avec décalage éventuel des poids lingots dans des lingotières trop grandes, n'est pas certaine. En effet, dans le cas d'un couple partitionné, il y a risque de vouloir affecter tous les lingots dans les plus grandes lingotières exclusivement.

Exemple :

Soit le stock de lingotières :

type	niveau Max	nombre
L1	85	10
L2	84	30
L3	83	15
L4	80	20
L5	78	40

Soit un couple traité en deux partitions dont les solutions partielles sont :

Solution 1 :	9 lingots de 85, 26 lingots de 84.	Solution 2 :	14 lingots de 83, 18 lingots de 80.
total :	35 lingots, 2 poids.	total :	32 lingots, 2 poids.

Total global au couple : 67 lingots répartis sur 4 poids.

Supposons fixé à 3 le nombre maximum de poids lingots admis; il faut uniformiser. Or c'est impossible dans ce cas précis car seules les combinaisons affectées aux lingots utilisés dans chaque partition ont été conservées. Des lingotières plus petites sont donc maintenant inaccessibles bien que disponibles.

Dans l'exemple présenté, les 40 lingotières de dimension 78 sont inaccessibles car aucun lingot n'y a été affecté. Pour uniformiser les poids lingots, seules les lingotières de dimension 80, 83, 84 et 85 sont disponibles. Il n'est donc pas possible d'affecter tous les lingots d'un de ces poids aux lingotières encore disponibles d'un autre de ces poids. En effet, quel que soit le poids dont on désire réaffecter les lingots, il ne reste pas suffisamment de lingotières accessibles restant disponibles parmi les autres poids. L'uniformisation est donc impossible.

Conditions pour que l'uniformisation soit rendue impossible :

La mise en défaut de la contrainte des lingotières n'a lieu que conjointement à trois conditions :

1. Le couple a été partitionné,
2. Les plus grandes lingotières du stock sont très utilisées,
3. Le nombre de poids lingots différents autorisé est limité.

Si l'une au moins de ces conditions n'est pas vérifiée, l'uniformisation est garantie :

1. n'est pas vérifiée $\Rightarrow$ l'uniformisation n'a pas lieu.
2. n'est pas vérifiée $\Rightarrow$ pas d'épuisement des plus grandes lingotières, l'uniformisation est possible.
3. n'est pas vérifiée $\Rightarrow$ l'uniformisation n'est pas nécessaire car le nombre maximum de poids lingots admis n'est pas dépassé.

En ce qui concerne l'usine d'Esch-Belval, les lingotières de tous les types sont en nombre illimité, à l'exception du type des lingotières les plus grandes. La condition 2 n'est donc pas vérifiée car il sera toujours possible d'uniformiser les poids lingots en les "plaçant" dans des lingotières en nombre illimité. La contrainte portant sur le stock des lingotières ne sera donc jamais mise en défaut.

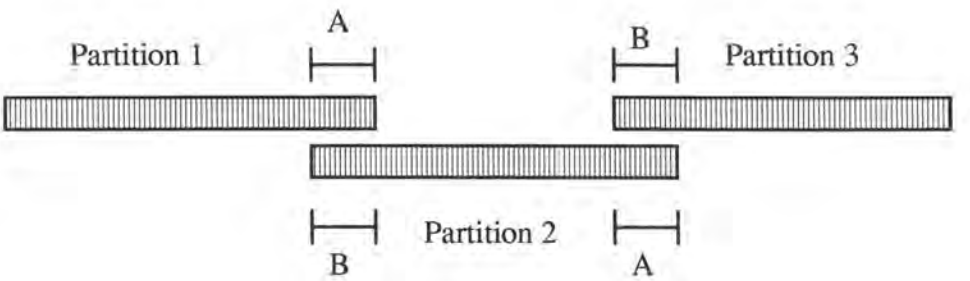
4. Chevauchement des partitions.

Concevoir un chevauchement des partitions offre deux avantages majeurs:

- ☞ transparence du partitionnement dans la génération des combinaisons

  le chevauchement des partitions permet de combiner des longueurs de poutrelles apparaissant dans chacune des partitions du couple en chevauchement, de sorte qu'il n'y ait pas de scission dans la génération des combinaisons.

- ☞ utilisation des poutrelles produites en surplus sur certaines parties en chevauchement (parties A):



*les poutrelles de chaque partition sont produites avec certitude, et éventuellement avec surplus (contrainte de recouvrement).

*les quantités de poutrelles à produire pour les longueurs en A doivent donc être réduites de moitié. Sinon, après production dans la première partition, il ne restera aucune poutrelle à produire en B, ce qui reviendrait à ne pas avoir de chevauchement.

*les quantités de poutrelles à produire pour les longueurs en B sont le total moins ce qui a déjà été produit dans la partition précédente. Si un surplus de production est apparu en A, il est donc déduit de ce qu'il reste à produire en B.

5. Gestion du chevauchement des partitions.

La gestion s'effectue à deux endroits:

1. dans les fichiers XPRESS pour le P.L.: modifier le nombre de poutrelles à produire pour les longueurs en A et B, du schéma ci-dessus.

2. lors de la génération des combinaisons: idem car la génération tient compte des quantités de poutrelles à produire, voir critère d'élimination des combinaisons, chap. IV.

Elle se déroule comme suit:

1. fichier P.L.:

   ∀ k indice d'une longueur de la partition,

   $Qtt\_à\_produire_k = Qtt\_totale_k - Qtt\_déjà\_produite_k$

   Pour les longueurs k faisant partie du chevauchement avec la partition suivante,

   $Qtt\_à\_produire_k = Qtt\_totale_k / 2$

2. génération des combinaisons:

   ∀ k indice d'une longueur dans la fenêtre de génération:

   $occur\_max = Qtt\_totale_k - Qtt\_déjà\_produite_k$

   Si la fenêtre de génération est la dernière sur la partition,

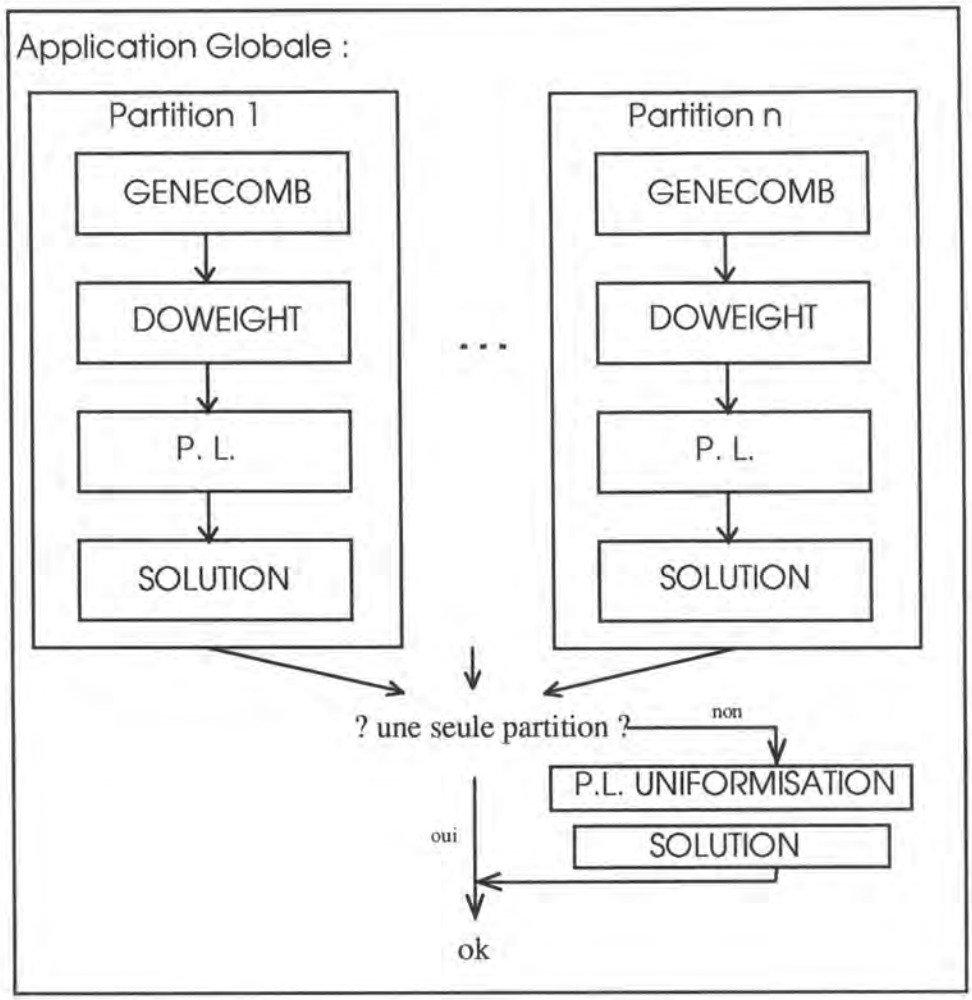
   et Si la partition n'est pas la dernière partition du couple

   Alors ∀ k indice d'une longueur de la fenêtre de génération tq cette longueur apparaît dans la partition suivante:

   $occur\_max = [Qtt\_totale_k / 2]$

6. L'Application Globale.

L'application globale prend en charge la gestion du partitionnement du couple traité. Sa structure est modifiée selon le schéma suivant :



VI. Spécifications de l'Application Globale

A. Le poids lingot d'une barre-mère.

Nous avons jusqu'à présent considéré les lingots en termes de longueurs de barres-mères. Ceci n'est pas le reflet de la réalité. Le poids lingot est lié à la longueur de la barre-mère obtenue par laminage par :

$$P = ((L + Chut\_Scies) * Pmét + Chut\_cisail) * (1 + Perte\_feu)$$

où P est le poids lingot, en kg.

L est la longueur totale utilisable de la barre-mère, en mètres.

Chut_Scies correspond aux chutages aux scies finisseuses, en mètres.

Pmét est le poids métrique, en kg/m.

Chut_cisail est le chutage aux cisailles, en kg.

Perte_feu est le pourcentage de perte au feu, $\in [0,1[$.

Cette relation entre les poids lingots établis par le P.L. (en mètres) et les poids lingots effectifs (en kilos) peut être effectuée dans le module Solution de l'Application Globale.

B. Les données et variables de l'Application Globale.

1. Les données en entrée de l'Application Globale.

L'application globale travaille au niveau d'un couple (dérivé,nuance) donné, de taille quelconque, pour lequel la commande d'acier correspondante est établie. Les données en entrée de l'application globale sont donc relatives à un couple (dérivé,nuance).

- Conventions d'écriture:

  les données sont positives,

  N: un entier, R: un réel,

  N* ou R*: tableau à une dimension,

  N** ou R**: tableau à deux dimensions,

  etc...

- Un couple (dérivé, nuance) décrit par:

Cple_NLong :	nombre de longueurs présentes dans le couple.	(N)
$Cple\_Long_k$:	les longueurs du couple, en mètres. k = 1..Cple_NLong	(R*)
$Cple\_Qtt_k$:	nombre de poutrelles à produire pour chacune des longueurs k du couple. k = 1..Cple_NLong	(N*)
2_ébauches :	Vrai / Faux selon le cas.	(V)
Pds_Métr :	poids métrique des barres-mères, en kg/m.	(R)

- Les données relatives aux barres-mères :

Lmin_BM :	longueur minimum, en mètres, des barres-mères admises à la découpe, tout chutage inclus.	(R)
Lmax_BM :	longueur maximum, en mètres, des barres-mères admises à la découpe, tout chutage inclus.	(R)
Pct_Fixe :	pourcentage statistique fixe de manque d'acier au laminoir vis-à-vis du tonnage commandé (typiquement 0,03). Ce manque est à prendre en compte une fois sur deux.	([0,1[)

- Les données relatives aux lingots:

Chut_scies :	chutage aux scies finisseuses, en mètres.	(R)
Chut_cisail :	chutage aux cisailles, en kg	(R)

Perte-feu : pourcentage de perte au feu
([0,1[)

Max_Pds : nombre maximum de poids lingots admis (N)

• Les données relatives aux lingotières:

N_Lingot : nombre de types de lingotières valides pour le couple (N)

NB_Lingot_l : nombre de lingotières de type l disponibles (N*)
l = 1..N_Lingot

Min_Lingot_l : tonnage minimum, en kg, des lingotières de type l (R*)
l = 1..N_Lingot

Max_Lingot_l : tonnage maximum, en kg, des lingotières de type l (R*)

2. Les variables à la sortie de l'Application Globale.

L'application globale fournit les données, pour la commande d'acier, relative au couple (dérivé, nuance) donné:

• Les poids lingots utilisés, et pour chacun d'eux, le nombre de lingots :

Cple_NWght : nombre de poids lingots utilisés (N)

$Cple\_Wght_j$: poids lingots utilisés, exprimés en m (R*)
j = 1..Cple_NWght

Cple_NBWghtj : nombre de lingots de poids Cple_Wghtj (N*)
j = 1..Cple_NWght

• L'affectation des lingots dans les lingotières:

$P\_Lingot_j^p$: n° du type de lingotière recevant la $p^{ème}$ occurrence du lingot de poids $Cple\_Wght_j$
(N**)

$p = 1..Cple\_NBWght_j$

j = 1..Cple_NWght

• Les combinaisons de découpe des lingots:

$BM\_Comb_{j,\,k}^p$: combinaison de découpe, en termes des coefficients pour les k long. du couple, de la première occurrence du lingot de poids $Cple\_Wght_j$ (N***)

k = 1..Cple_Nlong

$p = 1..Cple\_NBWght_j$

j = 1..Cple_NWght

==> on a donc l'affectation des barres-mères aux lingots.

3. Le partage des données inter-modules.

Les constantes:

CPLE_MAX_NLONG : Nbre maximum de longueurs de poutrelles traitables par l'application C.A.E.B. au sein d'un couple (derive,nuance).

PL_MAX_NLONG : Nbre maximum de longueurs de poutrelles traitables par le Programme Lineaire en une seule execution.

PL_MAX_COMB : Nbre maximum de combinaisons traitables par le P.L.

PL_WIN_SIZE : Taille de la fenetre sur les longueurs de poutrelles, i.e. nombre de coefficients par combinaisons.

OVERLAP : Taille du chevauchement lors du deplacement de la fenetre des combinaisons.

Les variables :

Gestion des donnees du couple complet:

CPLE_NLONG : Nbre total de longueurs dans le couple traite.

CPLE_LONG : Longueurs de poutrelles du couple traite, en metres.

CPLE_QTT : Nombre de poutrelles a produire pour chacune des longueurs du couple traite.

EBAUCHES : Nombres d'ebauches pour le couple traite.

LMIN_BM : Taille minimale des barres-meres, chutage scies inclus, en metres.

LMAX_BM : Taille maximale des barres-meres, chutage scies inclus, en metres.

PDS_METR : Poids metrique des barres-meres, en kg/metres.

CHUT_SCIES : Chutage des barres-meres aux scies finisseuses, en metres.

CHUT_SCISAIL : Chutage des lingots aux scisailles, en kg.

MAX_PDS : Nbre maximum de poids lingots differents pour la CAEB.

PCT_FIXE : Pourcentage statistique fixe de perte au laminoir. ([0;1[)

PERTE_FEU : Pourcentage de perte au feu des lingots. ([0;1[)

Gestion des resultats du couple complet:

CPLE_QOK : Nombre de poutrelles produites pour chacune des longueurs du couple traite.

CPLE_NCOMB : Nbre de combinaisons choisies par le P.L. pour le couple traite complet.

CPLE_NBCOMB : Nombre de fois que chaque combinaison choisie par le P.L. pour le couple complet est utilisee.

CPLE_COMB : Combinaisons choisies par le P.L. pour le couple complet.

CPLE_SHIFT : Valeur du deplacement des combinaisons CPLE_COMB sur les longueurs.

CPLE_LGCOMB : Longueur des combinaisons choisies par le P.L. pour le couple traite complet.

CPLE_NWGHT : Nbre de poids lingots choisis par le P.L. pour le couple

traite complet.

CPLE_NBWGHT : Nombre de fois que chaque poids lingot choisi par le P.L. est utilisee.

CPLE_WGHT : Poids choisis par le P.L. pour le couple traite complet.

USED : Somme, sur toutes les comb. generees admises pour le couple complet, des coeff. relatif a une longueur de poutrelle.

OBJECTIF : Valeur de la fonction objective pour le couple traite.

POURCENT : Pourcentage a ajouter pour compenser la perte de PCT_FIXE

Gestion des partitions:

PART_NLONG : Nombre de longueurs dans la partition courante du couple traite. (partition pour traitement par P.L.)

PART_NLONG = PART_LST_LG - PART_FRST_LG +1

PART_NLONG .LE. PL_MAX_NLONG

PART_FRST_LG : Indice de la premiere longueur de la partition courante dans les longueurs de poutrelles du couple traite.

PART_LST_LG : Indice de la dernire longueur de la partition courante dans les longueurs de poutrelles du couple traite.

NB_COMB : Nombre total de combinaisons generees dans la partition courante du couple traite.

COMB : Coefficients de toutes les combinaisons retenues au sein de la partition courante du couple traite.

LG_COMB : Longueurs des combinaisons COMB.

SHIFT : Valeur du deplacement des combinaisons COMB sur les longueurs.

NB_POIDS Nbre de poids lingots candidats pour la partition courante du couple traite.

POIDS : Poids lingots candidats a la CAEB pour la partition courante du couple traite.

Gstion de la fenetre sur les longueurs dans les partitions:

WIN_SIZE Taille effective de la fenetre sur les longueurs de poutrelles de la partition courante du couple traite, i.e. nombre de coefficients par combinaisons.

WIN_COMB : Nbre de combinaisons pouvant etre retenues lors de chaque deplacement de la fenetre sur les longueurs.

COEFF : Coefficients d'une combinaison generee.

LONG : Longueurs de poutrelles apparaissant dans la fenetre sur les longueurs lors de chaque deplacement.

QTT : Nombre de poutrelles a produire pour chacune des longueurs apparaissant dans la fenetre sur les longueurs.

C. Algorithme général.

Informellement, l'application globale de la commande d'acier se déroule comme suit:

- 1 lecture des données relatives au couple.
- 2 interprétation des données.

$$\text{Pourcent} \leftarrow \frac{1}{1 - Pct\_fixe} - 1$$

L_min_BM ← L_min_BM - Chut_scies

L_max_BM ← L_max_BM - Chut_scies

- 3 partitionnement du couple

calcul du nombre de partitions du couple

et de la taille des partitions: Part_NLong ←

$$\text{Min}\left(\text{Cple\_NLong} ; E\left[\frac{\text{Cple\_NLong}}{E\left[\frac{\text{Cple\_NLong}}{\text{PL\_max\_NLong - Overlap}}\right] + 1}\right] + 1 + \text{Overlap}\right)$$

- 4 Initialisation du couple complet

Cple_NComb ← 0

Cple_NWght ← 0

Objectif ← 0

$Cple\_QOK_k \leftarrow 0$ (quantité déjà produite)

Part_Frst_lg ← 1

Part_Lst_lg ← Part_NLong

- 5 Taille de la fenêtre sur les longueurs

$$\text{Win\_size} \leftarrow \begin{cases} \text{PL\_Win\_size si Part\_NLong} \geq \text{PL\_Win\_size} \\ \text{Part\_NLong sinon} \end{cases}$$

▪6 pour chaque partition:

NB_Comb ← 0

▪6.1calculer le nombre de déplacements de la fenêtre (taille partition/taille fenêtre).

▪6.2calculer le nombre de combinaisons pouvant être retenues par déplacement de la fenêtre(nb de comb tot retenues/nb de fenêtres).

$$Win\_Comb \leftarrow \left\lceil \frac{PL\_max\_Comb}{E\left[\frac{Part\_NLong}{Win\_size - Overlap}\right] + 1} \right\rceil$$

▪6.3pour chaque déplacement de la fenêtre:

Répéter

▪6.3.1 générer les combinaisons pour les longueurs de la fenêtre.

$Long_{1..Win\_size} \leftarrow Cple\_Long_{Part\_Frst\_lg..Part\_Frst\_lg+Win\_size-1}$

$Qtt_{1..Win\_size} \leftarrow Cple\_Qtt_{Part\_Frst\_lg..Part\_Frst\_lg+Win\_size-1}$

$Cple\_Qtt_{Part\_Frst\_lg..Part\_Frst\_lg+Win\_size-1}$

appeler GENECOMB(Win_size,Long,Qtt,Lmin_BM,Lmax_BM,coeff)

▪6.3.2 déplacer la fenêtre sur les longueurs de la partition courante:

mise à jour de Part_Frst_lg (indice de début de fenêtre)

Win_Comb ← Win_Comb0 (nb de comb par fenêtre)

Jusqu'à ce que Part_Frst_lg+Win_size-1=Part_Lst_lg

Part_Frst_lg ← Part_Lst_lg-Part_NLong+1

▪6.4générer les poids lingots sur base des combinaisons générées.

appel à DOWEIGHT

▪6.5écrire les fichiers de données du P.L. pour la partition courante.

LONG.XPR = $(Cple\_Long_k)_{k=Part\_Frst\_lg..Part\_Lst\_lg}$

NLONG.XPR = $(Cple\_Qtt_k)_{k=Part\_Frst\_lg..Part\_Lst\_lg}$

DIM.XPR = (Part_NLong, NB_Comb, NB_Poids, Max_Pds, Win_size, N_Lingot, Pct_fixe)

COMB.XPR = $(Comb(i,k)_{k=1..Win\_size})_{i=1..NB\_Comb}$

SHIFT.XPR = $(Shift_i)_{i=1..NB\_Comb}$

POIDS.XPR = $(Poids_j)_{j=1..NB\_Poids}$

LINGOT.XPR = $(Min\_Lingot_l, Max\_Lingot_l, NB\_Lingot_l)_{l=1..N\_Lingot}$

☠ NB_Lingot doit être corrigé au temps t pour pouvoir partitionner le couple

▪6.6 appel au P.L.

▪6.7 extraction de la solution et "append".

appel à SOLUTION

▪6.8 passage à la partition suivante.

(Part_Lst_lg < Cple_NLong) ⇒ ⌈ Part_Frst_lg ← Part_Lst_lg+1-Overlap
| Part_Lst_lg ←
| Part_Frst_lg+Part_NLong-1
⌇ corriger Part_Frst_lg et Part_Lst_lg
| si en dehors du couple
⌊ retour en ▪6

▪7 s'il y a plusieurs partitions:

▪7.1 générer les poids lingots sur base des combinaisons de la solution extraite.

Uniformiser les poids lingots si nécessaire

(Cple_NLong > PL_Max_NLong) ⇒ ⌈ appeler UNIFORM
| Objectif ← 0
⌇ Cple_NComb ← 0
| Cple_NWght ← 0
⌊ appeler SOLUTION

▪7.2 écrire les fichiers de données du P.L. pour le couple complet.

▪7.3 appel au P.L.

▪7.4 extraction de la solution et remplacement de la précédente.

▪8 Ecriture de la solution utilisateur:

Nombre de poids lingots: Cple_NWght

Poids lingots en tonnes: $[((Cple\_Wght_j + Chut\_scies)*Pds\_métr + Chut\_cisail) * (1+Perte\_feu) / 1000]_{j=1..Cple\_NWght}$

Quantités: $Cple\_NBWght_j$

Nombre de barres-mères: $\sum_{i=1}^{Cple\_NComb} Cple\_NBCombi$

Surplus de production: $Cple\_QOK_k - Cple\_Qtt_k$, $k=1..Cple\_NLong$

Combinaisons de découpe:

$(Cple\_Comb_{i,k}$ fois longueur$[Cple_Long_{k+shifti}])_{i=1..Cple_NComb}$

Lingotières: affecter chaque lingot à une lingotière selon l'algorithme établi dans SOLUTION

Affectation des barres-mères et des lingots selon l'algorithme.

▪9 FIN

D. Spécification complète de l'application globale et de ses modules.

1. L'application globale.

* son rôle:

Application principale de la commande d'acier d'Esch-Belval pour un couple (dérivé, nuance) donné.

Lecture des données utilisateurs, établissement de la commande d'acier par P.L., interprétation des résultats, partitionnement du couple.

* en entrée:

Les valeurs des variables suivantes sont lues dans les fichiers DIM.XPR, CPLE_LG.XPR, CPLE_QTT.XPR.

Cple_NLong: nombre de longueurs dans le couple

$Cple\_Long_k$: les longueurs

$Cple\_Qtt_k$: le nombre de poutrelles pour la longueur k

2_ébauches ; Lmin_BM ; Lmax_BM ; Pds_métr ; Chut_scies ; Chut_cisail ; Pct_fixe ; Max_Pds ; Perte_feu ; N_Lingot ; NB_Lingot_l ; Min_Lingot_l ; Max_lingot_l

* en sortie:

Cple_NWght ; $Cple\_NBWght_j$; $Cple\_Wght_j$

Cple_NComb ; $Cple\_Comb_i$; $Cple\_Shift_i$; $Cple\_NBComb_i$; $Cple\_LGComb_i$

affectation lingots -> lingotières

affectation comb -> lingots

* appelle:

GENECOMB → STOCOMB → ELIM → EVAL

DOWEIGHT

P.L.

SOLUTION

UNIFORM → P.L.

2. Le module GENECOMB.

* son rôle:

voir description.

* Paramètres:

Win_size: nombre de longueurs de poutrelles

$Long_k$: les longueurs

Qtt_k: nombre de poutrelles de longueur k à produire

Lmin_BM: longueur minimale des combinaisons

Lmax_BM: longueur maximale des combinaisons

$Coeff_k$: tableau de coefficients pouvant recevoir une combinaison

* en entrée

$used_k$:

Win_Comb: nombre maximum de combinaisons mémorisées

Part_Frst_lg: numéro de la 1ère longueur de la fenêtre (de la longueur de poutrelle $Long_1$ dans $Cple\_Long_k$)

NB_Comb

* en sortie:

$Win\_Comb = Win\_Comb_{entrée}$ - nombre de comb. mémorisées

$used_k$

$NB\_Comb = NB\_Comb_{entrée}$ + nombre de comb. mémorisées

$Comb_{i,k}$

$shift_i$

Lg_Comb_i

* appelle:

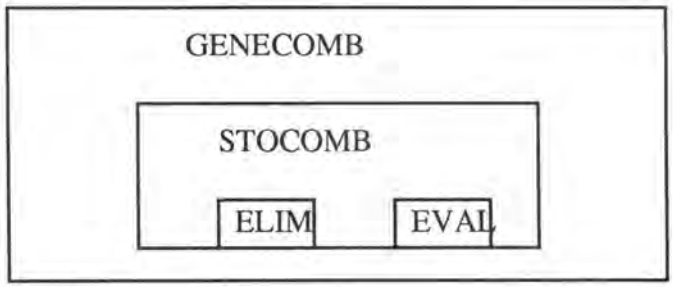
STOCOMB → ELIM → EVAL

* appelé par:

Application globale

* structure:

GENECOMB est spécifié comme suit



GENECOMB génère les combinaisons, une par une

A chaque combinaison générée, appel au module STOCOMB pour la mémorisation éventuelle.

STOCOMB mémorise les combinaisons reçues de GENECOMB après validation par appel au module ELIM et pondération par appel au module EVAL

Les combinaisons sont mémorisées dans un tableau ayant une dimension maximum (nombre de combinaisons retenues) trié selon l'ordre croissant des pondérations.

ELIM valide une combinaison proposée par l'intermédiaire de STOCOMB par renvoi d'un booléen d'acceptation ou de rejet de la combinaison en fonction de critères d'élimination.

EVAL évalue une combinaison proposée par l'intermédiaire de STOCOMB par renvoi d'une pénalité qui est la somme des pénalités attribuées pour chaque critère d'évaluation.

Algorithme

- 0 k'=1
- 1 établir tous les coefficients d'une combinaison

$$\forall\ k=k'..Win\_size:\ coeff_k=\min(Cple\_Qtt_k,[l/Cple\_Long_k])$$

$$\text{avec } l = L\max\_BM - \sum_{a=1}^{k-1}(coeffa * Cple\_Longa)$$

- 2 assurer que comb ≥ Lmin_BM

zéro = ⌈0 si $\alpha \geq$ Lmin_BM

⌊E[(Lmin_BM-α)/coeff$_{Win_size}$]+1 sinon

avec $\alpha = \sum_{k=1}^{Win\_size-1} (coeff_k * Cple\_Long_k)$

▪3 écrire la combinaison et les suivantes

(coeff$_k \geq$ zéro) $\Rightarrow$ ⌈STOCOMB(|comb|)

{coeff$_{Win_size}$--

⌊retour en ▪3

▪4 la combinaison suivante exige que coeff$_{Win_size-1}$ soit modifié

▪5 (k' $\exists$) $\Rightarrow$ ⌈coeff$_{k'}$--

{k'++

⌊retour en ▪1

▪6 stop

3. Le module STOCOMB.

* son rôle:

voir description

* Paramètres:

Leff: longueur de la combinaison reçue (= | Comb$_i$ |)

* en entrée:

Win_size ; Coeff$_k$; used$_k$; Part_Frst_lg ; Qtt$_k$; Win_Comb ; NB_Comb

* en sortie:

NB_Comb ; Win_Comb ; Comb$_i$; shift$_i$; Lg_Comb$_i$; used$_i$

* appelle:

ELIM

EVAL

* appelé par:

GENECOMB

Algorithme:

- si la combinaison n'est pas rejetée par ELIM(coeff) faire:
  - EVAL(coeff)
  - chercher où coeff (i.e. la combinaison) doit être insérée dans le tableau trié des combinaisons: COMB
  - si insertion possible, alors
    * NB_Comb++
    * $Comb_{NB\_Comb,k} \leftarrow coeff_k$
    * $shift_{NB\_Comb} \leftarrow Part\_Frst\_lg - 1$
    * $Lg\_Comb_{NB\_Comb,k} \leftarrow Leff$
    * mettre à jour Used
    * Win_Comb--

4. Le Module ELIM.

* son rôle:

voir description

5. Le module EVAL.

* son rôle:

voir description.

* en entrée:

$coeff_k$

* en sortie:

la pondération calculée: EVAL = $\sum$ pondérations(critère)

6. Le module DOWEIGHT.

* son rôle:

voir description.

* en entrée:

Cple_NComb ; Lg_Comb_i ; $shift_i$; Pourcent ; Ecart ; données des lingotières

* en sortie:

NB_Poids ; Poids

* appelé par:

Application Globale

UNIFORM

7. Le module UNIFORM.

* son rôle:

Uniformisation des solutions des P.L. exécutés pour chaque partition du couple. L'uniformisation consiste en la réexécution du P.L., sur le couple complet cette fois, sur base des combinaisons choisies pour les différentes partitions par les précédentes exécutions du P.L.

* en entrée::

Cple_NComb, Cple_Comb, CpleLgComb, Cple_Shift

* en sortie:

NB_Comb, NB_Poids, Poids, Lg_Comb, Shift, Comb

* appelé par:

l'application globale

* appelle:

le P.L.

DOWEIGHT

Conclusions

1. Le logiciel de Programmation Linéaire XPRESS-MP.

Le logiciel XPRESS-MP de Dash Associates utilisé pour supporter la modélisation et l'optimisation du Programme Linéaire s'est avéré efficace à plus d'un point de vue :

- Facilité d'utilisation :

  XPRESS-MP comporte un modéliseur, un optimiseur et un rapporteur. Ceux-ci s'utilisent assez aisément par emploi de quelques commandes appropriées clairement spécifiées dans le manuel d'utilisation. Le modéliseur est à la fois simple et puissant au niveau de la sémantique mathématique supportée. L'optimiseur est très paramétrable et assez puissant. Le rapporteur permet d'extraire du fichier solution des variables les valeurs de variables spécifiées.

- Souplesse de l'optimiseur :

  L'optimiseur offre une extension spécifique aux problèmes linéaires en variables entières : le IP-Opt (Integer Programming Optimiser). Le IP-Opt peut être paramétré dans sa politique de recherche d'une valeur optimale (Branch & Bond, priorité des variables, ...).

- Diversité des types de variables :

  L'Integer Programming Optimiser supporte des variables de type binaire (binary variables), entières (integer variables), partiellement entières (partial integer variables), semi-continues (semi-continuous variables), SOS1 et SOS2 (Special Order Set of type 1 & 2). Des variables de type SOS1 sont toutes nulles sauf au plus une d'entre-elles. Des variables de type SOS2 sont toutes nulles sauf au plus deux d'entre-elles qui dans ce cas sont consécutives dans leur ordre d'indice.

- Puissance :

  Différents tests ont montré que les premières solutions entières proposées par l'IP-Opt sont en général une excellente approche de la solution optimale entière. Ces solutions apparaissent plus ou moins rapidement en fonction de la taille du modèle. Concrètement, dans le cas de notre modèle linéaire les premières solutions entières apparaissent endéans quelques minutes. Des solutions optimales sont parfois atteintes au bout de 10 à 30 minutes. Il arrive cependant fréquemment que le modèle tourne plusieurs heures sans que ne soit trouvée de solution optimale. Dans ce cas, on peut remarquer que la fonction objective n'est pas améliorée de façon très significative au fur et à mesure des solutions entières intermédiaires. La solution optimale n'est pas absolument indispensable. Une solution satisfaisant les exigences de l'entreprise semble en général rapidement atteinte.

• Multi-plateformes :

Le logiciel XPRESS-MP est proposé pour des plateformes fonctionnant sous MS-DOS, OS/2, VMS, UNIX. Des fichiers de type Lotus 1-2-3, Symphony et DBase peuvent être interprétés pour constituer les données d'un modèle.

2. La Programmation Linéaire comme outil industriel.

La théorie mathématique de la programmation linéaire présente des avantages certains. Bien que le temps mis pour trouver une solution optimale reste imprévisible, des applications de plus en plus nombreuses peuvent être envisagées sur base de cette théorie mathématique au vu de la puissance logicielle et matérielle actuellement disponibles sur le marché. L'application mise en oeuvre pour la Commande d'Acier en est exemple concret. Je pense que la programmation linéaire utilisée en tant qu'outil informatique dans un contexte industriel a un avenir très prometteur.

3. Apport du stage.

Le stage effectué à ARBED Luxembourg a été pour moi très enrichissant. Sur le plan des connaissances, par l'étude et l'utilisation approfondie du logiciel XPRESS-MP ainsi que par l'étude du langage FORTRAN avec lequel l'application de la Commande d'Acier a été conçue. Sur le plan de l'expérience également, par la réalisation d'un projet important en collaboration avec des informaticiens et non informaticiens. Sur le plan technique enfin, par le travail effectué sur différents sites informatiques.

Annexes

- **L'aspect temps réel de la Commande d'Acier.**
- **Code Fortran de l'Application Globale.**

- **L'aspect temps réel de la Commande d'Acier.**

DECOUPE EN TEMPS REEL
A LA SCIE GREY
ARBED DIFFERDANGE

Analyse de faisabilité

PREETUDE

27 Octobre 1992

Jean-Pierre CASTIAUX

Marc LESCRENIER

ARBED Administration Centrale

Service de l'Informatique
Unité Modélisations et Optimisations

TABLE DES MATIERES

I. INTRODUCTION

I.1. Avertissement.

Ce document ne consiste qu'en une préétude de la faisabilité de la découpe en temps réel des barres-mères et ne se veut donc en aucun cas être exhaustif.

Le but de la préétude se limite à l'identification de l'intéret d'une automatisation de la découpe.

Notamment, le facteur temps -prépondérant dans cette application- n'a pas été pris en compte.

I.2. Rappel succint de l'énoncé.

Suite à la commande d'acier pour un extrait de commandes donné (par abus de langage, nous dirons extrait) vient la découpe des barres-mères en temps réel. Cette découpe est en principe conditionnée par la planification établie lors de la commande d'acier.

Mais la production de barres-mères est sujette à de nombreux facteurs perturbateurs difficilement contrôlables, de sorte que les barres-mères reçues à la découpe diffèrent très fréquemment de celles attendues au vu de la commande d'acier.

C'est pourquoi une aide à la découpe en temps réel est nécessaire. Il s'agit de déterminer la découpe de chaque barre-mère se présentant aux scies. La détermination de la découpe d'une barre-mère est entre autres fonction de ses caratéristiques physiques, des poutrelles restant à produire dans l'extrait, et des contraintes d'expéditions.

I.3. D'une heuristique à une optimisation plus globale.

Une solution envisageable, à la fois simple et probablement efficace, est la suivante: étant donnée une barre-mère à découper -appelée barre-mère courante-, la découpe choisie est la plus grande combinaison des longueurs de poutrelles restant à produire qui lui soit affectable.

Une combinaison de longueurs de poutrelles est dite affectable à une barre-mère si elle peut "y entrer" et respecte toutes les contraintes à ce moment précis.

Les contraintes à respecter doivent bien sûr être définies. Nous les laissons de côté ici.

Cette solution semble acceptable. Le chutage est minimisé à chaque découpe. Il s'agit donc d'une minimisation *locale* du chutage.

Une autre solution envisageable est de ne pas restreindre à un phénomène local le choix de découpe de la barre-mère courante, mais au contraire de l'étendre à un phénomène plus *global*, c'est-à-dire en tâchant de considérer plusieurs barres-mères à venir (dont la première est la barre-mère courante) dans le choix de la combinaison de découpe de la barre-mère courante.

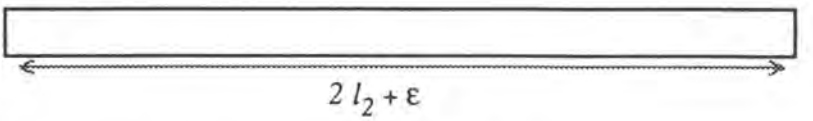
Un exemple simple montre en effet que cette solution -que nous appellerons solution globale- peut être beaucoup plus efficace que la première solution, dite locale:

Considérons l'état d'un extrait quelconque, à un moment donné, et supposons les données suivantes:

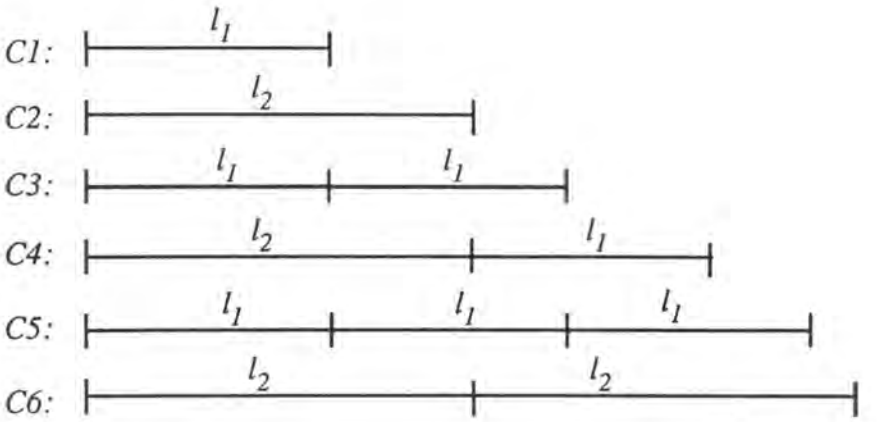
Longueurs des poutrelles à produire: l_1 , l_2

Nombre de poutrelles restant à produire: $2 * l_1$, $1 * l_2$

Barre-mère courante:



Combinaisons des longueurs (maximum: $2 * l_2$):

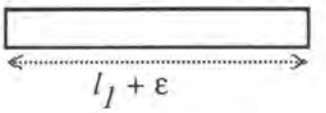


Le choix d'une combinaison pour la découpe de la barre-mère donne pour résultat:

Comb. de découpe	Chutage	Stock	Reste à produire
C6	0	$1 * l_2$	$2 * l_1$
C5	faible	$1 * l_1$	$1 * l_2$
C4	important	0	$1 * l_1$

La méthode locale dicte de choisir C6 (ou éventuellement C5) comme combinaison de découpe, mais rejettera C4 car il résulterait de son utilisation un chutage important et aucun ajout au stock.

Or ce serait la combinaison à utiliser si la barre-mère suivante arrivant aux scies est:



La solution locale, elle, requiert une troisième barre-mère pour recouvrir l'extrait.

La solution globale quant à elle revient à choisir une combinaison de découpe de la barre-mère de sorte qu'il sera possible de produire l'extrait restant avec un minimum d'acier.

Il s'agit donc de replanifier la production des poutrelles restant à produire dans l'extrait chaque fois qu'une barre-mère se présente aux scies, et d'en déduire sa combinaison de découpe.

Cette planification sera réalisée sur base des barres-mères que l'on s'attend à recevoir, c'est-à-dire respectivement la barre-mère courante, les barres-mères en attente déjà prêtes pour la découpe et les barres-mères commandées lors de la commande d'acier. Il va de soi que l'efficacité de cette solution dépend directement du degré de certitude de la réception des longueurs des barres-mères arrivant aux scies à la suite de la barre-mère courante.

Cette solution étant une solution en temps réel, elle sera itérative.

Résumé schématique:

DECOUPE EN TEMPS REEL

DECOUPER LA BARRE-MERE RECUE AUX SCIES.
(BARRE-MERE COURANTE)

TROUVER UNE COMBINAISON DE DECOUPE DE LA BARRE-MERE COURANTE EN S'ASSURANT QUE L'ON SAURA PRODUIRE TOUT L'EXTRAIT AVEC UN MINIMUM D'ACIER

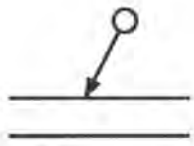
PLANIFIER LA DECOUPE DE TOUTES LES POUTRELLES RESTANT À PRODUIRE DANS L'EXTRAIT AU VU DES BARRES-MERES SUIVANT LA BARRE-MERE COURANTE

II. PREMIERE ANALYSE: LE FLUX DES DONNEES.

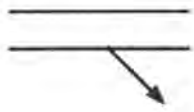
Pour rappels:

- un extrait de commandes est reçu en une séquence ordonnée de postes à produire.
- à ce stade, l'extrait de commandes est vu comme une partition ordonnancée de couples (dérivé, nuance).
- étant donné un couple (dérivé, nuance), on connaît le nombre de postes inclus, le nombre de poutrelles à produire pour chacun des postes du couple et les données relatives à leur expédition et a leur chargement.
- par abus de langage, les mots "postes" et "longueurs de poutrelles" sont utilisés indifféremment.

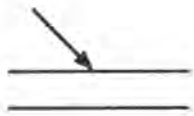
Légende:



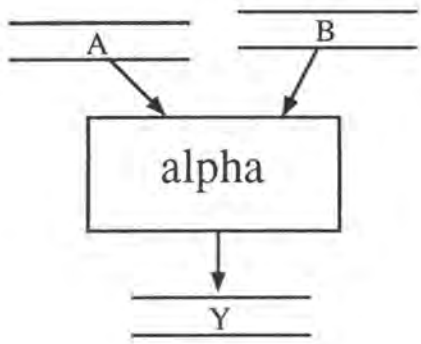
Donnée initiale.



Donnée consultée.

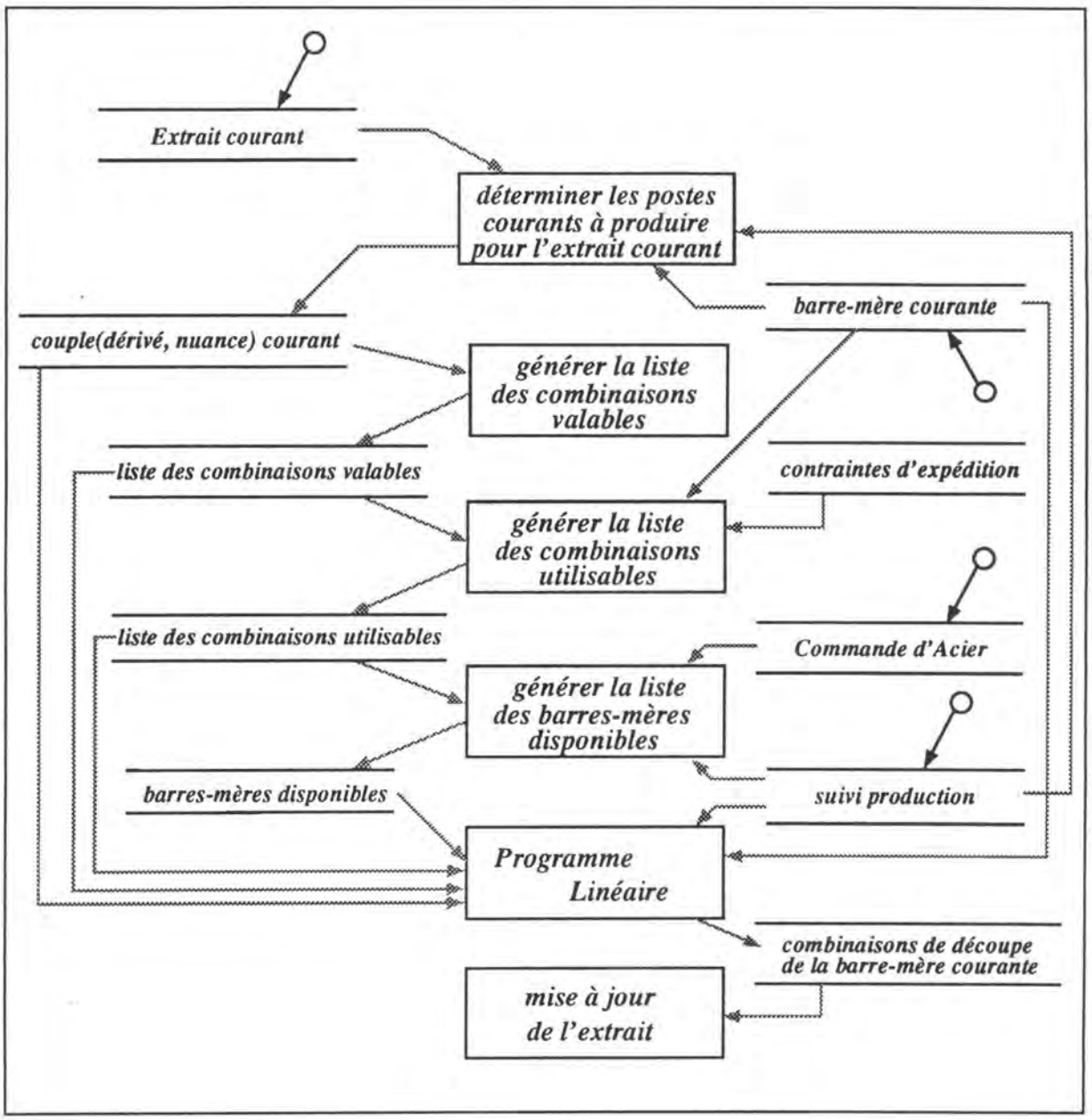


Donnée produite.



Le module alpha produit les données Y SSI les données consultées A et B existent non vides.

Dynamique d'une découpe (cfr fonctionnalités pour détails):



III. SECONDE ANALYSE: FONCTIONNALITES

Les différents modules inhérents au flux des données méritent quelques explications. Notons cependant que la description ci-dessous est indépendante des considérations pratiques d'implémentation et de temps d'exécution. Chaque module devra par la suite être analysé en particulier.

déterminer les postes courants à produire pour l'extrait courant

- INPUT:
  - Extrait ordonnancé à produire = séquence de couples (dérivé, nuance).
- OUTPUT:
  - Couple courant.
- EXECUTION:
  - Pointeur vers le couple courant dans l'extrait, en fonction du suivi de production.
  - Déterminer les postes restant à produire pour le couple correspondant à la barre-mère courante.

mise à jour de l'extrait

- INPUT:
  - Extrait ordonnancé à produire.
- OUTPUT:
  - Extrait ordonnancé restant à produire.
- EXECUTION:
  - Soustraire du couple courant les poutrelles produites par la barre-mère courante.

générer la liste des combinaisons valables

- INPUT:
  - Couple courant.
- OUTPUT:
  - Liste de toutes les combinaisons *valables* pour le couple courant
- EXECUTION:
  - Si les postes à produire ont changés, regénérer les combinaisons valables.

générer la liste des combinaisons utilisables

- INPUT:
  - Caractéristiques de la barre-mère courante.
  - Combinaisons *valables*.
  - Postes concernés par chacune des combinaisons valables.
  - Etat du chargement.
- OUTPUT.
  - Combinaisons *utilisables*.
- EXECUTION:
  - Pour chacune des combinaisons valables, détermination de la valeur d'une variable 0/1 indiquant si la combinaison peut/ne peut pas être utilisée pour le chargement au moment de l'exécution.

générer la liste des barres-mères disponibles

- INPUT:
  - La Commande d'Acier.
  - Le suivi production (barres-mères certaines, déjà prêtes pour la découpe).
  - Les combinaisons utilisables.
- OUTPUT:
  - Les barres-mères supposées arriver tôt ou tard aux scies.
- EXECUTION:
  - Les barres-mères supposer arriver aux scies sont de deux ordres:
    . les barres-mères certaines, dont la barre-mère courante
    . les barres-mères attendues au vu de la Commande d'Acier
  - Les barres-mères supposées arriver aux scies seront les barres-mères certaines et les barres-mères "déduites" de la Commande d'Acier en fonction de facteurs divers inhérents au temps réel.
  - Plusieurs politiques de "déduction" des barres-mères à partir de la Commande d'Acier sont possibles, en fonction du respect de la séquence des barres-mères dans la Commande d'Acier, ou du choix possible des barres-mères en son sein.

- Si le choix des barres-mères dans la Commande d'Acier est admis, les barres-mères "déduites" peuvent peut-être être celles qui s'approchent le plus des combinaisons utilisables afin de tenir compte le mieux possible de l'état du chargement.

Programme Linéaire

- INPUT:
  - Les barres-mères supposées disponibles y compris les barres-mères certaines dont la barre-mère courante.
  - Les données relatives au couple à produire.
  - Les combinaisons *valables*.
  - Les combinaisons *utilisables*.

- OUTPUT:

  - *La* combinaison de découpe de la barre -mère courante.

- EXECUTION:

  - Détermination parmi les combinaisons utilisables de la meilleure combinaison de découpe de la barre-mère courante en fonction des barres-mères supposées venir aux scies, et de l'état courant du couple à produire.

  - cfr analyse détaillée ci-après.

IV. L'OPTIMISATION A CHAQUE ETAPE:

PROGRAMMATION LINEAIRE.

IV.1. But.

Le but du programme linéaire est essentiellement de deux ordres:

- replanification de la découpe des barres-mères (c-à-d sélection des combinaisons utilisées) en fonctions des barres-mères attendues (toutes ou les quelques suivantes) et du couple à recouvrir (ou des quelques postes suivants).
- en particulier, détermination de la meilleure combinaison pour la découpe de la barre-mère courante.

IV.2. Données.

ENTREES:

- K: n^{bre} de types de longueurs de poutrelles (postes) à produire.
- I: n^{bre} de types de combinaisons de découpe.
- J: n^{bre} de types de barres-mères supposées disponibles.
- les types de longueurs de poutrelles à produire:

$$l_k,\ k = 1..K,\ K > 0$$

- le couple à produire (n^{bre} de poutrelles à produire par longueur):

$$XT_k,\ k = 1..K$$

- les longueurs des types de barres-mères disponibles avec certitude, dont la barre-mère à découper (barre-mère courante):

$$L_j,\ j = 1..m,\ m > 0$$

- les longueurs des types de barres-mères supposées disponibles ultérieurement:

$$L_j,\ j = 1..J,\ J \geq m$$

- le nombre de barre-mères types de L_j consécutives:

$$N_j,\ j = 1..J$$
$$N_j = 1,\ j = 1..m$$

- les combinaisons de longueurs de poutrelles *valables*[1]:

$$COMB_i,\ i = 1..I,\ I > 0$$
$$COMB_i \equiv \left(COMB_{i,k}\right)_{k=1..K}$$

$COMB_{i,k}$: *coeff. de la k° longueur de poutrelle de la comb. i*

- les combinaisons de longueurs de poutrelles *utilisables*[2].

$$COMB_i \ tq\ USABLE_i = 1$$
$$USABLE_i : \textit{données } 0 \ / \ 1 : \textit{comb}_i \textit{ est / n'est pas utilisable},\ i = 1..I$$

SORTIES:

- *la* combinaison de découpe de la barre-mère courante.

IV.3. Description des contraintes.

- La planification recouvre complètement les poutrelles à produire et est réalisable.

  complètement: en effet, les éventuelles barres-mères supplémentaires à la commande d'acier sont inclues dans la séquence des barres-mères attendues aux scies.

- L'acier est utilisé en quantité minimale.

  But poursuivi, toute contrainte par ailleurs considérée.

- Une des combinaisons utilisables est utilisée pour la découpe de la barre-mère courante.

- Les barres-mères certaines sont utilisées.

  Cette contrainte représente l'objet même de la découpe en temps réel: produire les poutrelles commandées à partir des barres-mères arrivant aux scies.

  Rem: la barre-mère courante est la seule barre-mère certaine en fin de production du couple.

IV.4. Modélisation des contraintes.

- La planification recouvre les poutrelles à produire complètement et est réalisable.

  recouvrir complètement:

  - variables $NCOMB_i$: nombre de fois que la combinaison i est utilisée.

[1] combinaison *valable*: combinaison pouvant servir à la découpe d'une barre-mère au sens des contraintes les plus générales.

[2] combinaison *utilisable*: combinaison valable pouvant servir à la découpe de la barre-mère courante au sens des contraintes dynamiques inhérentes au temps réel (chargement, expédition, ...).

- contraintes:

$$\forall\ k = 1..K:\ \sum_{i=1..I}\left(NCOMB_i\ *\ COMB_{i,k}\right)\ \geq\ XT_k$$

- Une des combinaisons utilisables est utilisée pour la découpe de la barre-mère courante.

découpe de la barre-mère avec comb. i?

- <u>variables binaires</u> δ_i: la comb. i (n')est (pas) utilisée pour la barre-mère courante.
- une seule combinaison pour la barre-mère courante:

$$\sum_{i=1..I}\delta_i = 1$$

- la combinaison choisie pour la barre-mère courante est utilisable:

$$\forall\ i = 1..I:\ \delta_i\ \leq\ USABLE_i$$

- comptabiliser la combinaison choisie pour la barre-mère courante:

$$\forall\ i = 1..I:\ \delta_i\ \leq\ NCOMB_i$$

<u>découper le reste</u>:

- le problème d'affectation des autres combinaisons utilisées dans les autres barres-mères a une solution

$$\Updownarrow$$

$$\forall\ i = 1..I:\ \sum_{\|\ COMB_l\|\ \geq\ \|\ COMB_i\|}\left(NCOMB_l - \delta_l\right)\ \geq\ \sum_{\substack{L_j \geq\ \|\ COMB_i\|\\ L_j \neq BM.\ courante}} N_j$$

$$avec\ \|COMBi\| = \sum_{k=1..K}\left(COMB_{i,k}\ *\ l_k\right)$$

- *Les barres-mères certaines sont utilisées.*

- <u>variables</u> $NINGOT_j$: nombre de fois que le type de barre-mère L_j est utilisé.

$$\forall\ j = 1..m:\ NINGOT_j\ \geq\ 1$$

- Acier utilisé en quantité minimale.

- fonction objective.

$$\text{minimiser} : \sum_{j=1..J} \left(NINGOT_j * L_j \right)$$

- Les barres-mères utilisées sont disponibles.

$$\forall j = 1..J: NINGOT_j \leq N_j$$

IV.5. Proposition d'une contrainte supplémentaire - critique.

Cette modélisation décrite peut bien sûr être affinée afin d'en améliorer le temps d'exécution ou de tenir compte d'autres contraintes et mieux représenter la réalité.

Notamment, les contraintes suivantes méritent d'être discutées:

- Utiliser le plus possible de combinaisons utilisables dans la planification de découpe.

Cette contrainte reflète en réalité le phénomène de continuité locale dans le chargement des wagons: une combinaison utilisable pour la découpe de la barre-mère courante le sera encore très probablement pour la découpe de la barre-mère suivante.

Elle n'est cependant pas facile à ajouter dans le modèle linéaire:

proposition modélisée par
- *objectif* ⇒ *P.L. multicritère* !
- *contrainte* ⇒ *borne inférieure* ?

- Choix de la fonction objectif.

Cette modélisation ne garantit pas le gain d'une barre-mère. Il serait intéressant de pouvoir indiquer une préférence pour des chutages concentrés au sein de quelques barres-mères uniquement. Ce modèle ne semble pas le permettre car l'affectation des combinaisons aux barres-mères n'est pas connue, afin d'en diminuer la complexité.

V. LIMITES DU PROGRAMME LINEAIRE.

Il est attendu de ce programme linéaire qu'il soit très efficace. Il ne faut cependant pas oublier que son efficacité repose sur la validité des données fournies et qu'il est donc indispensable de soigner particulièrement l'analyse de chacun des modules extérieurs, ainsi que leur interaction.

Il subsistera néanmoins deux critères en défaveur de cette solution par programmation linéaire:

- le temps d'exécution est variable, indéterminable, et exponentiel.
- le programme linéaire est très rapidement saturé par le nombre de variables à traiter.

Néanmoins, le second critère peut dans un premier temps facilement être contourné par l'utilisation de la notion de fenêtres. Notons qu'une amélioration portant sur le second critère a une conséquence directe favorable sur le premier.

Si le nombre de variables du programme linéaire est réduit, ce dernier ne fonctionnera que mieux. Il est donc intéressant de considérer les variables selon une vue restreinte à la dimension de fenêtres portant sur le nombre de barres-mères supposées disponibles et sur le nombre de postes (longueurs de poutrelles) à produire.

Ces choix se justifient aisément: une fenêtre sur les barres-mères disponibles permet de restreindre le facteur d'incertitude à seulement quelques barres-mères. Plus le nombre de barres-mères supposées arriver aux scies est grand, plus l'incertitude est grande. Remarquons cependant que cette fenêtre doit être suffisament large pour permettre de produire tous les postes k, k=1..K. Il est en outre important qu'elle contienne un maximum de barres-mères certaines. Quant à la fenêtre sur les postes à produire, elle permet de refuser de produire des postes trop éloignés à partir d'une même barre-mère et d'assurer ainsi que l'ordonnancement préalable de ces postes ne soit pas trop pertubé.

Le programme linéaire est lui totalement indépendant de cette notion de fenêtre. C'est à l'application lui fournissant les données que revient la gestion de ces fenêtres.

Mais la programmation linéaire peut rester insatisfaisante vis-à-vis de la contrainte de temps d'exécution, primordiale dans toute application en temps réel.

Par ce fait même, il peut être intéressant de se pencher à nouveau sur la solution locale décrite précédemment.

L'heuristique de détermination de la combinaison de découpe pourrait peut-être facilement être améliorée de façon à imiter au mieux la solution globale.

A titre d'exemple, elle pourrait éventuellement avantageusement être remplacée par la suivante:

"La combinaison de découpe choisie est celle qui, si utilisée, produit la plus petite somme de chutage et de longueurs des poutrelles qui resteront à produire."

La combinaison de découpe pourrait ainsi être déterminée en un seul parcours séquentiel du fichier des combinaisons *utilisables*, et ce choix ne serait plus mis en défaut par l'exemple introduit précédemment.

VI. CONCLUSION.

En conclusion de cette préétude, nous pouvons considérer que la découpe en temps réel des barres-mères peut être automatisée.

Une automatisation de la découpe engendrerait probablement un gain non négligeable, à la fois direct, par la minimisation du nombre de barre-mères utilisées dans la production d'un extrait de commandes, et indirect, par un meilleur respect des contraintes dynamiques de chargement et d'expédition.

Le problème de la conception d'une telle automatisation réside principalement dans la contrainte portant sur le temps maximum endéans lequel la réponse doit parvenir, ainsi que sur la difficulté de déterminer précisément quelles sont les barres-mères qui se présenteront aux scies à la suite de la barre-mère courante.

Plusieurs solutions sont proposées. Il importe de les tester sur base de données réelles afin de pouvoir comparer les résultats obtenus avec ceux actuellement établis.

- **Code Fortran de l'Application Globale.**

Scénario d'exécution.

Le scénario d'exécution est ici présenté dans une phase de test qui reflète l'état d'avancement du projet au moment de la fin du stage. L'Application Globale implémentée est celle détaillée dans le document à l'exception de la prise en charge de la contrainte des lingotières et de la contrainte des deux ébauches. L'Application Globale a été conçue pour fonctionner sur une plateforme VAX de Digital Equipment Corporation. Les données relatives au couple à traiter arrivent sous forme de fichiers destinés à l'Application Globale sur le site VAX. Le logiciel mathématique de programmation linéaire fonctionne lui dans sa version provisoire sur un PC 80486 DX. Pour lancer l'Application Globale, il faut donc :

1. lancer l'Application Globale sur VAX : 'run orchestr'
2. lorsque l'A.G. le demande, lancer XPRESS-MP sur le PC : 'run.bat'
3. les fichiers résultats du P.L. sont automatiquement transférés de PC vers le VAX
4. continuer l'exécution de l'A.G. quand le P.L. est terminé : '<ENTER>'
5. recommencer les opérations 2, 3 et 4 pour chaque partition du couple
6. si le couple a été partitionné, lancer le P.L. d'uniformisation sur PC : 'u_run.bat', les fichiers résultats sont automatiquement transférés du PC sur le VAX.
7. affichier la solution globale finale écrite dans le fichier CAEB.SOL sur VAX : 'ed caeb.sol'

Programmes BATCH pour MicroSoft-DOS.

RUN.BAT :

```
@echo off
echo Deleting *.xpr
del *.xpr
echo.
echo *.XPR files transfert from VAX to PC
copy f:\caeb\essai\*.xpr
edit long.xpr
edit nlong.xpr
edit comb.xpr
edit shift.xpr
edit poids.xpr
echo.
echo Running L.P.
call xpressmp.bat %1
echo.
echo Transfert of L.P. solution from PC to VAX
copy caeb.xpr f:\caeb\essai
echo Done.
```

XPRESSMP :

```
@echo off
if %1#==# set name=ca_13
if not %1#==# set name=%1
set name=\caeb\vax\%name%

echo.
echo Loading model %name%.mod
cd \xpressmp
lpmod387 %name% @\caeb\vax\run_mod.xlp

echo.
echo Deleting CAEB.*
del \caeb\vax\caeb.*

echo.
echo Running optimiser for model %name%.mod
echo    PLEASE  WAIT...
if %1# == uniform# goto uniform
ipopt387 %name% @\caeb\vax\run_opt.xlp>NUL
goto cont
:uniform
ipopt387 %name% @\caeb\vax\u_run_op.xlp
:cont

echo.
echo Reporting solution into file CAEB.XPR
reprt387 %name%<\caeb\vax\run_rprt.xlp>\caeb\vax\caeb.xpr
cd \caeb\vax

set name=
echo ok.
```

U_RUN.BAT :

```
@echo off
echo Deleting *.xpr
del *.xpr
echo.
echo *.XPR files transfert from VAX to PC
copy f:\caeb\essai\U_*.xpr
EDIT U_DIM.XPR
edit u_lcomb.xpr
edit u_ncomb.xpr
edit u_poids.xpr
echo.
echo Running L.P.
call xpressmp.bat uniform
echo.
echo Transfert of L.P. solution from PC to VAX
copy caeb.xpr f:\caeb\essai
echo Done.
```

RPT.BAT :

```
@echo off
if %1#==# set name=ca_13
if not %1#==# set name=%1
set name=\caeb\vax\%name%

cd \xpressmp
echo.
echo Reporting solution into file CAEB.XPR
reprt387 %name%<\caeb\vax\run_rprt.xlp>\caeb\vax\caeb.xpr
cd \caeb\vax

set name=
echo.
echo Transfert of L.P. solution from PC to VAX
copy caeb.xpr f:\caeb\pc486
echo Done.
```

L'Application Globale : ORCHESTR.FOR

Le partage des données inter-modules :

```
C                    ======================
C                      COMMON  C.A.E.B
C                    ======================


C===============
C  CONSTANTES
C===============


C     CPLE_MAX_NLONG: Nbre maximum de longueurs de poutrelles
C                     traitables par l'application C.A.E.B. au sein d'un
C                     couple (derive,nuance).
C     PL_MAX_NLONG:   Nbre maximum de longueurs de poutrelles traitables
C                     par le Programme Lineaire en une seule execution.
C     PL_MAX_COMB:    Nbre maximum de combinaisons traitables par le P.L.
C     PL_WIN_SIZE:    Taille de la fenetre sur les longueurs de poutrelles,
C                     i.e. nombre de coefficients par combinaisons.
C     OVERLAP:        Taille du chevauchement lors du deplacement de la
C                     fenetre des combinaisons.
C     PCT_FIXE:       Pourcentage statistique fixe de perte au laminoir.


      INTEGER*2   CPLE_MAX_NLONG, PL_MAX_NLONG, PL_MAX_COMB,
     >            PL_WIN_SIZE, OVERLAP

      PARAMETER   (CPLE_MAX_NLONG=200, PL_MAX_NLONG=25)
      PARAMETER   (PL_MAX_COMB=350, PL_WIN_SIZE=4, OVERLAP=2)
```

```
      REAL       PCT_FIXE

      PARAMETER  (PCT_FIXE=0.03000)


C===============
C  VARIABLES
C===============


C     CPLE_NLONG:  Nbre total de longueurs dans le couple traite.
C     CPLE_LONG:   Longueurs de poutrelles du couple traite.
C     CPLE_QTT:    Nombre de poutrelles a produire pour chacune
C                  des longueurs du couple traite.
C     CPLE_QOK:    Nombre de poutrelles produites pour chacune
C                  des longueurs du couple traite.
C     CPLE_NCOMB:  Nbre de combinaisons choisies par le P.L. pour le couple
C                  traite complet.
C     CPLE_COMB:   Combinaisons choisies par le P.L. pour le couple complet.
C     CPLE_SHIFT:  Valeur du deplacement des combinaisons CPLE_COMB sur les
C                  longueurs.
C     CPLE_NBCOMB: Nombre de fois que chaque combinaison choisie par le P.L.
C                  pour le couple complet est utilisee.
C     CPLE_LGCOMB: Longueur des combinaisons choisies par le P.L.
C                  pour le couple traite complet.
C     CPLE_NWGHT:  Nbre de poids lingots choisis par le P.L. pour le couple
C                  traite complet.
C     CPLE_NBWGHT: Nombre de fois que chaque poids lingot choisi par le P.L.
C                  est utilisee.
C     CPLE_WGHT:   Poids choisis par le P.L. pour le couple traite complet.
C     PART_NLONG:  Nombre de longueurs dans la partition courante
C                  du couple traite. (partition pour traitement par P.L.)
C                  PART_NLONG = PART_LST_LG - PART_FRST_LG +1
C                  PART_NLONG .LE. PL_MAX_NLONG
C     PART_FRST_LG: Indice de la premiere longueur de la partition courante
C                  dans les longueurs de poutrelles du couple traite.
C     PART_LST_LG: Indice de la dernire longueur de la partition courante
C                  dans les longueurs de poutrelles du couple traite.
C     LONG:        Longueurs de poutrelles apparaissant dans la fenetre sur
C                  les longueurs lors de chaque deplacement.
C     QTT:         Nombre de poutrelles a produire pour chacune des
C                  longueurs apparaissant dans la fenetre sur les longueurs.
C     WIN_SIZE:    Taille effective de la fenetre sur les longueurs de
C                  poutrelles de la partition courante du couple traite,
C                  i.e. nombre de coefficients par combinaisons.
C     WIN_COMB:    Nbre de combinaisons pouvant etre retenues lors de
C                  chaque deplacement de la fenetre sur les longueurs.
C     NB_COMB:     Nombre total de combinaisons generees dans la partition
C                  courante du couple traite.
C     COEFF:       Coefficients d'une combinaison generee.
C     COMB:        Coefficients de toutes les combinaisons retenues au sein
C                  de la partition courante du couple traite.
C     LG_COMB:     Longueurs des combinaisons COMB.
C     SHIFT:       Valeur du deplacement des combinaisons COMB sur les
C                  longueurs.
C     USED:        Somme, sur toutes les comb. generees admises pour le
```

```
C             couple complet, des coeff. relatif a une longueur de
C             poutrelle (compte non tenu des recouvrements).
C     LMIN_BM:    Taille minimale des barres-meres (i.e. des comb.).
C     LMAX_BM:    Taille maximale des barres-meres (i.e. des comb.).
C     MAX_PDS:    Nbre maximum de poids lingots differents pour la CAEB.
C     NB_POIDS:   Nbre de poids lingots candidats pour la partition
C             courante du couple traite.
C     POIDS:      Poids lingots candidats a la CAEB pour la partition
C             courante du couple traite.
C     POURCENT:   Pourcentage a ajouter pour compenser la perte de PCT_FIXE
C     OBJECTIF:   Valeur de la fonction objective pour le couple traite


      COMMON /COUPLE/
     >            CPLE_NLONG, CPLE_LONG, CPLE_QTT, CPLE_QOK,
     >            CPLE_LGCOMB, CPLE_NCOMB, CPLE_NBCOMB,
     >            CPLE_WGHT, CPLE_NWGHT, CPLE_NBWGHT,
     >            CPLE_COMB, CPLE_SHIFT, OBJECTIF

      INTEGER*2  CPLE_NLONG, CPLE_QTT(CPLE_MAX_NLONG),
     >          CPLE_QOK(CPLE_MAX_NLONG), CPLE_NCOMB,
     >          CPLE_COMB(PL_MAX_NLONG*20,PL_WIN_SIZE),
     >          CPLE_SHIFT(PL_MAX_NLONG*20),
     >          CPLE_NBCOMB(PL_MAX_NLONG*20),
     >          CPLE_NWGHT, CPLE_NBWGHT(PL_MAX_NLONG)

      REAL      CPLE_LONG(CPLE_MAX_NLONG),
     >          CPLE_LGCOMB(PL_MAX_NLONG*20),
     >          CPLE_WGHT(PL_MAX_NLONG), OBJECTIF


      COMMON /PARTITION/
     >            PART_NLONG, PART_FRST_LG, PART_LST_LG

      INTEGER*2  PART_NLONG, PART_FRST_LG, PART_LST_LG


      COMMON /GEN_COMB/
     >            LONG, QTT, WIN_SIZE, WIN_COMB, NB_COMB, COEFF,
     >            COMB, LG_COMB, SHIFT, LMIN_BM, LMAX_BM, USED

      INTEGER*2  QTT(PL_WIN_SIZE), WIN_SIZE, WIN_COMB, NB_COMB,
     >          COEFF(PL_WIN_SIZE), COMB(PL_MAX_COMB,PL_WIN_SIZE),
     >          SHIFT(PL_MAX_COMB), USED(CPLE_MAX_NLONG)

      REAL      LONG(PL_WIN_SIZE), LG_COMB(PL_MAX_COMB),
     >          LMIN_BM, LMAX_BM


      COMMON /WEIGHTS/
     >            MAX_PDS, NB_POIDS, POIDS, POURCENT

      INTEGER*2  MAX_PDS, NB_POIDS

      REAL      POIDS(2*PL_MAX_COMB), POURCENT
```

Le module principal ORCHESTR.FOR:

```
      PROGRAM CAEB

C
C ABSTRACT:
C
C     Etablir la commande d'acier pour un couple provenant d'un extrait
C     de commande.
C     Orchestration des differents modules interagissant dans la CAEB.
C
C INPUTS:
C
C   Un couple (derive,nuance), decrit selon les fichiers suivants:
C
C     Fichier DIM.XPR: (respectivement:)
C
C         - # longueurs dans le couple complet.
C         - longueur minimale des BM.
C         - longueur maximale des BM.
C         - # maximum de poids lingots differents admis pour la CA.
C
C     Fichier CPLE_LG.XPR:
C
C         - valeur en metres des longueurs du couple complet,
C           une longueur par ligne de fichier, dans l'ordre de
C           l'ordonnancement des postes.
C
C     Fichier CPLE_QTT.XPR:
C
C         - nombre de poutrelles pour chacune des longueurs du
C           couple complet, un nombre par ligne de fichier, dans l'ordre
C           des longueurs du fichier CPLE_LONG.XPR.
C
C OUTPUTS:
C
C     Fichier CAEB.RSL:
C
C         Commande d'acier du couple complet.
C
C BODY:
C
C     Voir analyse.
C
C AUTHOR(S):
C
C     J-P Castiaux
C
C CREATION DATE:
C
C     11-12-92
C
C MODIFICATION HISTORY:
C
C     Date    | Name | Description
```

```
C ---------------+------+-------------------------------------------------
C               |      |
C [change_entry]


C ================
C   GLOBAL
C ================

      IMPLICIT NONE
      INCLUDE 'ORCHESTR.BLK'


C ================
C  VARIABLES
C ================

C     I1, K:      Variable integer de travail.

      INTEGER*2 I1, K, OVERLAP0, WIN_COMB0
      REAL   MITRAILLE, STOCK, TO_BE_DELETED


C ================
C  MAIN
C ================


C     *ABOUT...*

      WRITE(*,*)
      WRITE(*,*) '      ORCHESTRATION  C.A.E.B'
      WRITE(*,*) '      ======================='
      WRITE(*,*)


C     *LECTURE DES DIMENSIONS DU COUPLE*

100   FORMAT (1X,A30,I3.1)
110   FORMAT (1X,A39,F7.3,' metres.')
120   FORMAT (1X,A40,I3.1)
      OPEN(10,FILE='DIM.XPR',STATUS='OLD')
      READ(10,*) CPLE_NLONG, LMIN_BM, LMAX_BM, MAX_PDS
      CLOSE(10)
      WRITE(*,100) ' Nombre total de longueurs:   ', CPLE_NLONG
      WRITE(*,110) ' Longueur minimale des barres-meres:   ', LMIN_BM
      WRITE(*,110) ' Longueur maximale des barres-meres:   ', LMAX_BM
      WRITE(*,120) ' Nombre maximum de poids lingots admis: ', MAX_PDS
      IF (CPLE_NLONG .GT. CPLE_MAX_NLONG) THEN
     WRITE(*,*)
     WRITE(*,*) ' -> ERREURS DE DONNEES DANS DIM.XPR! - NOMBRES
     >TROP GRANDS!'
     GOTO 99999
      ENDIF


C     *CALCUL DU POURCENTAGE FIXE A AJOUTER*
```

```
      POURCENT = 1.0/(1.0-PCT_FIXE) - 1.0
      POURCENT = INT(POURCENT*1E5+1.0) / 1.0E5
125   FORMAT (1X,A40,F10.8)
      WRITE(*,125) ' % statistique fixe de perte:          ',PCT_FIXE
      WRITE(*,125) ' % fixe de compensation a ajouter:     ',POURCENT
      WRITE(*,*)


C     *LECTURE DES LONGUEURS ET QUANTITES*

      OPEN(10,FILE='CPLE_LG.XPR',STATUS='OLD')
      READ(10,*) (CPLE_LONG(K), K=1,CPLE_NLONG)
      CLOSE(10)
      OPEN(10,FILE='CPLE_QTT.XPR',STATUS='OLD')
      READ(10,*) (CPLE_QTT(K), K=1,CPLE_NLONG)
      CLOSE(10)
      WRITE(*,*) ' Extrait:'
      WRITE(*,*)
127   FORMAT (10X,'Longueur ',I2.2,':  ',F7.3,' metres.   (',
     >        I4.4,'x)')
      WRITE(*,127) ((K,CPLE_LONG(K),CPLE_QTT(K)), K=1,CPLE_NLONG)
      WRITE(*,*)


C     *PARTITION DU COUPLE*

C     *Nbre de longueurs par partition*
      IF (OVERLAP.GE.CPLE_NLONG) THEN
     OVERLAP0=0
      ELSE
     OVERLAP0 = OVERLAP
      ENDIF
      I1 = CPLE_NLONG/(PL_MAX_NLONG-OVERLAP0)
      IF (I1*(PL_MAX_NLONG-OVERLAP0).LT.CPLE_NLONG-OVERLAP0) THEN
     I1 = I1 + 1
      ENDIF
      PART_NLONG = CPLE_NLONG/I1
      IF (PART_NLONG*I1.LT.CPLE_NLONG-OVERLAP0) THEN
     PART_NLONG = PART_NLONG + 1
      ENDIF
      PART_NLONG = MIN(PART_NLONG + OVERLAP0 , CPLE_NLONG)

C     *Initialisation*
      PART_FRST_LG = 1
      PART_LST_LG = PART_NLONG
      CPLE_NCOMB=0
      CPLE_NWGHT=0
      OBJECTIF=0
      DO 1050, I1=1,CPLE_NLONG
     CPLE_QOK(I1)=0
1050  CONTINUE

C     *Ecrire les longueur de la partition courante*
1020  IF (PART_NLONG .NE. CPLE_NLONG) THEN
      WRITE(*,*)
      WRITE(*,*) ' ---------- PARTITION ----------'
      WRITE(*,*)
```

```
      WRITE(*,*) ' Longueurs traitees:'
      WRITE(*,*)
      WRITE(*,127) ((K,CPLE_LONG(K),CPLE_QTT(K)),
     >              K=PART_FRST_LG,PART_LST_LG)
      WRITE(*,*)
       ENDIF


C     *FENETRE SUR LA PARTITION COURANTE*

       WIN_SIZE = PL_WIN_SIZE
       IF (PART_NLONG.LT.PL_WIN_SIZE) THEN
      WIN_SIZE = PART_LST_LG-PART_FRST_LG+1
       ENDIF


C     *GENERATION DES COMBINAISONS POUR LA PARTITION COURANTE*
      .
       NB_COMB = 0
       WIN_COMB = PART_NLONG/(WIN_SIZE-OVERLAP0)
       IF (WIN_COMB*(WIN_SIZE-OVERLAP0).LT.PART_NLONG-OVERLAP0) THEN
      WIN_COMB=WIN_COMB+1
       ENDIF
       WIN_COMB = PL_MAX_COMB/WIN_COMB
       WIN_COMB0 = WIN_COMB
1010   DO 1000, I1=1,WIN_SIZE
      LONG(I1) = CPLE_LONG(PART_FRST_LG+I1-1)
      QTT(I1) = MAX(CPLE_QTT(PART_FRST_LG+I1-1) -
     >              CPLE_QOK(PART_FRST_LG+I1-1) ,0)
      USED(PART_FRST_LG+I1-1)=0
1000   CONTINUE
       IF (PART_FRST_LG+WIN_SIZE-1.EQ.PART_LST_LG .AND.
     >   PART_LST_LG.LT.CPLE_NLONG) THEN
C        *qtt(lg recouvertes)=cple_qtt/2*
      DO 1060, I1=PART_LST_LG-OVERLAP0+1,PART_LST_LG
        QTT(I1-PART_FRST_LG+1) = CPLE_QTT(I1)/2
1060     CONTINUE
       ENDIF
       CALL GENECOMB(WIN_SIZE,LONG,QTT,LMIN_BM,LMAX_BM,COEFF)
       WIN_COMB=WIN_COMB0
       IF (PART_FRST_LG+WIN_SIZE-1.LT.PART_LST_LG) THEN
      PART_FRST_LG = PART_FRST_LG + WIN_SIZE - OVERLAP0
      IF (PART_FRST_LG+WIN_SIZE-1.GT.PART_LST_LG) THEN
        PART_FRST_LG = PART_LST_LG - WIN_SIZE +1
      ENDIF
      GOTO 1010
       ENDIF
       WRITE(*,*) ' Nombre de combinaisons generees: ',NB_COMB
       WRITE(*,*)
       PART_FRST_LG = PART_LST_LG - PART_NLONG +1


C     * GENERATION DES POIDS LINGOTS CANDIDATS *

       CALL DOWEIGHT


C     *CREATION DES FICHIERS XPRESS*
```

```
C    *LONG.XPR*
     OPEN(10,FILE='LONG.XPR',STATUS='UNKNOWN')
133  FORMAT(1X,F7.3)
     WRITE(10,133) (CPLE_LONG(K), K=PART_FRST_LG,PART_LST_LG)
     CLOSE(10)

C    *NLONG.XPR*
     OPEN(10,FILE='NLONG.XPR',STATUS='UNKNOWN')
135  FORMAT(1X,I4.4)
     WRITE(10,135) (MAX(CPLE_QTT(K)-CPLE_QOK(K),0),
    >           K=PART_FRST_LG,PART_LST_LG-OVERLAP)
     IF (PART_LST_LG.LT.CPLE_NLONG) THEN
C       *qtt(lg recouvertes)=cple_qtt/2*
    WRITE(10,135) ((CPLE_QTT(K)/2),
    >           K=PART_LST_LG-OVERLAP+1,PART_LST_LG)
     ELSE
    WRITE(10,135) ((CPLE_QTT(K)),
    >           K=PART_LST_LG-OVERLAP+1,PART_LST_LG)
     ENDIF
     CLOSE(10)

C    *DIM.XPR*
     OPEN(10,FILE='DIM.XPR',STATUS='UNKNOWN')
130  FORMAT (I2.2,',',I3.3,',',I3.3,',',I2.2,',',I2.2)
     WRITE(10,130) PART_NLONG, NB_COMB, NB_POIDS, MAX_PDS, WIN_SIZE
     CLOSE(10)

C    *COMB.XPR*
     OPEN(10,FILE='COMB.XPR',STATUS='UNKNOWN')
140  FORMAT (1X,<WIN_SIZE-1>(I2.2,','),I2.2)
141  FORMAT ('-',<WIN_SIZE-1>(I2.2,','),I2.2)
     WRITE(10,140) (COMB(1,K), K=1,WIN_SIZE)
     DO 1030, I1=2,NB_COMB
    WRITE(10,141) (COMB(I1,K), K=1,WIN_SIZE)
1030 CONTINUE
     CLOSE(10)

C    *SHIFT.XPR*
     OPEN(10,FILE='SHIFT.XPR',STATUS='UNKNOWN')
150  FORMAT(1X,I2.2)
     DO 1040, I1=1,NB_COMB
   WRITE(10,150) SHIFT(I1)
1040 CONTINUE
     CLOSE(10)

C    *POIDS.XPR*
160  FORMAT (1X,F8.4)
C170  FORMAT ('-',<NB_POIDS>(8X,F8.4))
     OPEN(10,FILE='POIDS.XPR',STATUS='UNKNOWN')
     WRITE(10,160) (POIDS(I1), I1=1,NB_POIDS)
     CLOSE(10)
     WRITE(*,*) ' Nombre de poids lingots candidats:',NB_POIDS
C     WRITE(*,170) (POIDS(I1), I1=1,NB_POIDS)


C    * APPEL AU PROGRAMME LINEAIRE *
```

```
      WRITE(*,*) '--------- TAPER "0 <ENTER>" APRES
     >TERMINAISON DU P.L. ---'
      READ(*,*) TO_BE_DELETED
      WRITE (*,*) 'OK.'


C     * EXTRACTION DE LA SOLUTION PARTIELLE*

      CALL SOLUTION


C     * PARTITION SUIVANTE DU COUPLE*

      IF (PART_LST_LG.LT.CPLE_NLONG) THEN
     PART_FRST_LG = PART_LST_LG +1 - OVERLAP0
     PART_LST_LG = PART_FRST_LG + PART_NLONG -1
     IF (PART_LST_LG.GT.CPLE_NLONG) THEN
       PART_LST_LG = CPLE_NLONG
       PART_NLONG = PART_LST_LG - PART_FRST_LG +1
     ENDIF
     GOTO 1020
      ENDIF


C     * UNIFORMISATION DES POIDS LINGOTS SI COUPLE PARTITIONNE*

      IF (CPLE_NLONG.GT.PL_MAX_NLONG) THEN
     WRITE(*,*)
     WRITE(*,*) '---------- UNIFORMISATION DES POIDS LIMGOTS -------'
     WRITE(*,*)
     CALL UNIFORM
     OBJECTIF = 0
     CPLE_NWGHT = 0
     CALL U_SOLUTN
      ENDIF


C     * EXTRACTION DE LA SOLUTION COMPLETE*

      OPEN(90,FILE='CAEB.RSL',STATUS='UNKNOWN')
      WRITE(90,*)
      WRITE(90,*)
      WRITE(90,*) '=================='
      WRITE(90,*) 'SOLUTION GLOBALE:'
      WRITE(90,*) '=================='
      WRITE(90,*)
      WRITE(90,*) ' Production:'
      WRITE(90,*)
 200  FORMAT (10X,I4.3,' x Longueur ',I2.2,':',F7.3,' metres.   (+',
     >       I4.1,')')
      WRITE(90,200) ((CPLE_QOK(K),K,CPLE_LONG(K),
     >              CPLE_QOK(K)-CPLE_QTT(K)), K=1,CPLE_NLONG)

C      *Stock*
      STOCK=0
      DO 1080, K=1,CPLE_NLONG
     STOCK=STOCK+(CPLE_QOK(K)-CPLE_QTT(K))*CPLE_LONG(K)
 1080  CONTINUE
```

```
202   FORMAT (33X,'(=',F9.3,' metres.)')
      WRITE(90,202) STOCK

C     *Acier*
      WRITE(90,*)
      WRITE(90,*)
180   FORMAT (1X,' Longueur d''acier effective:',F10.3,' metres.')
      WRITE(90,180) OBJECTIF
      WRITE(90,*)
      WRITE(90,*) ' Stock total: ',STOCK,' metres.'

C     *Mitraille*
      MITRAILLE = OBJECTIF
      DO 1070, K=1,CPLE_NLONG
     MITRAILLE=MITRAILLE-CPLE_QOK(K)*CPLE_LONG(K)
1070  CONTINUE
190   FORMAT (1X,' Mitraille (lg. eff. - prod.): ',F10.3,' metres.')
      WRITE(90,*)
      WRITE(90,190) MITRAILLE

C     *Lingots*
      WRITE(90,*)
      WRITE(90,*) ' Commande de lingots:'
      WRITE(90,*)
210   FORMAT (10X,I4.1,' x POIDS',I2.2,' : ',F8.5,' metres')
      I1=0
      DO 1090,K=1,CPLE_NWGHT
     WRITE(90,210) CPLE_NBWGHT(K),K,CPLE_WGHT(K)
     I1 = I1 + CPLE_NBWGHT(K)
1090  CONTINUE

C     *Barres-Meres*
      WRITE(90,*)
220   FORMAT (1X,' Nombre de barres-meres:  ',I3.1)
      WRITE(90,220) I1
      WRITE(90,*)
230   FORMAT (1X,' Nombre de combinaisons de decoupe:  ',I3.1)
      WRITE(90,230) CPLE_NCOMB
      WRITE(90,*)

C     *combinaisons de decoupe*
      WRITE(90,*)
      WRITE(90,*) ' COMBINAISONS DE DECOUPE:'
      WRITE(90,*) ' ----------------------'
      WRITE(90,*)
240   FORMAT(10X,I3.1,' x COMB.',I3.3,':',
     >      <WIN_SIZE>(' ',I2.1,'xL',I2.2))
      DO 1100,I1=1,CPLE_NCOMB
     WRITE(90,240) CPLE_NBCOMB(I1),I1,
     >             ((CPLE_COMB(I1,K),CPLE_SHIFT(I1)+K),K=1,WIN_SIZE)
1100  CONTINUE
      WRITE(90,*) '(',CPLE_NCOMB,'comb. de decoupe )'

      CLOSE(90)

99999 END
```

Le module GENECOMB.FOR

```
      SUBROUTINE GeneComb(NBRE_LG,LONG,QTT,LMIN0,LMAX0,COEFF)

C
C ABSTRACT:
C
C       Generer toutes les combinaisons de n longueurs donnees,
C       chaque longueur pouvant apparaitre plusieurs fois dans
C       une meme combinaison.
C
C
C INPUTS:
C
C     Parametres recus:
C
C       NBRE_LG:     Nombre total de longueurs donnees.
C       LONG(i):     Les longueurs donnees.
C       QTT(i):      Occurence maximale de la longueur i dans les comb.
C       LMIN0:       Longueur minimale des combinaisons.
C       LMAX0:       Longueur maximale des combinaisons.
C
C
C OUTPUTS:
C
C     Les combinaisons sont envoyees une a une a la sous-routine STOCOMB,
C     supposee exister correcte, via les parametres (dans l'ordre):
C
C       NBRE_LG: voir inputs.
C       COEFF(i, i=1..NBRE_LG):  Occurence de la longueur i dans la comb.
C       LEFF: Longueur de la combinaison.
C
C
C BODY:
C
C     Les combinaisons generees sont telles que:
C
C          - 0 .LE. COEFF(i) .LE. QTT(i),  pour i=1..NBRE_LG
C          - SUM( COEFF(i) * LONG(i) ) .LE. LMAX0, i=1..NBRE_LG
C          - SUM( COEFF(i) * LONG(i) ) .GE. LMIN0, i=1..NBRE_LG
C
C
C AUTHOR(S):
C
C       J-P Castiaux
C
C CREATION DATE:
C
C       26-09-92
C
C MODIFICATION HISTORY:
C
C       Date     | Name  | Description
C ---------------+-------+-----------------------------------------------
C  [change_entry] |       |
```

```
      IMPLICIT NONE

      INTEGER*2 NBRE_LG,
     >          QTT(NBRE_LG), COEFF(NBRE_LG),
     >          I, ZERO, K
      REAL      LONG(NBRE_LG), LMIN0, LMAX0, LMAX, Q



C ==========
C  INIT
C ==========

      LMAX = LMAX0
      I = 1
      ZERO = 0

C ===============
C  COMBINAISONS
C ===============

C     *Etablir tous les coeff.*
10    COEFF(I) = MIN(QTT(I), INT(LMAX/LONG(I)))
      IF (I.NE.NBRE_LG) THEN
20       LMAX = LMAX - COEFF(I)*LONG(I)
     I = I+1
     GOTO 10
      ENDIF

C     *S'assurer que la comb. est aussi grande que LMIN0*
      IF (LMAX0-LMAX.LT.LMIN0) THEN
     Q = (LMIN0-LMAX0+LMAX)/LONG(I)
     IF (ABS(Q-INT(Q)).LT.1E-8) THEN
       ZERO = INT(Q)
     ELSE
       ZERO = INT(Q)+1
     ENDIF
      ENDIF

C     *Ecrire la combinaison*
C      (test pour eviter un cas d'erreur, voir analyse)
      IF (COEFF(I).GE.ZERO) THEN
     CALL STOCOMB(LMAX0-LMAX+COEFF(I)*LONG(I))
      ENDIF

C     *La combinaison suivante est la meme avec COEFF(NBRE_LG) decremente*
30    IF (COEFF(I).GT.ZERO) THEN
     COEFF(I) = COEFF(I)-1
     CALL STOCOMB(LMAX0-(LMAX-COEFF(I)*LONG(I)))
     GOTO 30
      ENDIF

C     *La comb. suiv. exige que le coeff precedent soit modifie*
      ZERO = 0
40    I = I-1
```

```
C    *Reiterer le processus*
     IF (I.NE.0) THEN
    LMAX = LMAX + COEFF(I)*LONG(I)
    IF (COEFF(I).LE.ZERO) THEN
      GOTO 40
    ELSE
      COEFF(I) = COEFF(I)-1
      GOTO 20
    ENDIF
     ENDIF

C    *Generation terminee*
     RETURN
     END
```

Le module STOCOMB.FOR

```
C-------------------------------------------------------------------------
C  ROUTINE CHARGEE DE STOCKER LA COMBINAISON PASSEE DANS LE TABLEAU DES
COMB.
C-------------------------------------------------------------------------
  SUBROUTINE STOCOMB(LEFF)

  IMPLICIT NONE
  INCLUDE 'ORCHESTR.BLK'

  INTEGER*2 I, I1, I2, OK
  REAL LEFF


C------- # DE LG DANS LA COMB.

  OK=0
C     DO 10, I=1,WIN_SIZE
C       IF (COEFF(I).NE.0) THEN
C         OK=OK+1
C       ENDIF

C------- MAXIMUM DES COEFF.

C       IF (COEFF(I).GT.3) THEN
C         OK=20
C       ENDIF
10    CONTINUE

C------- MAXIMUM DE COEFF. #1

C     IF (COEFF(1).GT.3) THEN
C       OK=20
C     ENDIF

C------- VARIATION DES COEFF.

C     DO 20, I=1,WIN_SIZE-1
```

```
C       IF (COEFF(I).GT.COEFF(I+1)+2 .AND. COEFF(I+1).NE.0) THEN
C         OK=20
C       ENDIF
C20   CONTINUE

C------- RAPPORT ENTRE COEFF CONSECUTIFS.

  I1=0
  DO 35, I=1,WIN_SIZE
    IF (USED(PART_FRST_LG+I-1).EQ.0 .AND.
   >      COEFF(I).NE.0) THEN
      I1=I1+1
    ENDIF
35    CONTINUE
  IF (I1.EQ.0) THEN
    DO 40, I=1,WIN_SIZE-1
      IF ( COEFF(I)*COEFF(I+1).NE.0 ) THEN
            IF (COEFF(I).GT.COEFF(I+1)) THEN
              I1=I
              I2=I+1
            ELSE
              I2=I
              I1=I+1
            ENDIF
            IF ( COEFF(I1)/COEFF(I2) .LT. QTT(I1)/QTT(I2) ) THEN
              OK=20
            ENDIF
            IF ( COEFF(I1)/COEFF(I2) .GT. QTT(I1)/QTT(I2)+1 ) THEN
              OK=20
            ENDIF
      ENDIF
40      CONTINUE
  ENDIF

C------- SAUVEGARDE

  IF (OK.LE.4 .AND. WIN_COMB.NE.0) THEN
    NB_COMB = NB_COMB + 1
    WIN_COMB = WIN_COMB-1
    DO 30, I=1,WIN_SIZE
      COMB(NB_COMB,I) = COEFF(I)
30      CONTINUE
    SHIFT(NB_COMB) = PART_FRST_LG - (PART_LST_LG - PART_NLONG +1)
    LG_COMB(NB_COMB) = LEFF
    DO 50, I=1,WIN_SIZE
      USED(PART_FRST_LG+I-1) = USED(PART_FRST_LG+I-1) + COEFF(I)
50      CONTINUE
  ENDIF

  RETURN
  END
```

Le module DOWEIGHT.FOR

```
      SUBROUTINE DoWeight

C
C ABSTRACT:
C
C     Etablir les poids lingots candidats a la commande d'acier pour
C     le Programme Lineaire de CAEB.
C
C BODY:
C
C     Les poids lingots candidats sont les longueurs des combinaisons de
C     poutrelles, ainsi que ces longueurs augmentees du pourcentage
C     statistique fixe de perte au laminoir.
C     Les candidats superieurs (inferieurs) a la longueur maximale
C     (minimale) de barre-mere precisee par l'utilisateur sont rejetes.
C     Les candidats egaux a un ecart pres precise par l'utilisateur sont
C     consideres comme identiques.
C     Les candidats retenus sont uniques.
C
C INPUTS:
C
C     CPLE_NCOMB, CPLE_LCOMB
C
C OUTPUTS:
C
C     NB_POIDS, POIDS
C
C AUTHOR(S):
C
C     J-P Castiaux
C
C CREATION DATE:
C
C     23-09-92
C
C MODIFICATION HISTORY:
C
C      Date   | Name  | Description
C ------------+-------+-----------------------------------------------------
C   23-09-92  |DOWEIGHT| Noyau du programe: longueurs des combinaisons.
C   12-11-92  |       | + unicite + regroupement.
C   17-11-92  |       | + longueur maximale de barre-mere.
C   19-11-92  |       | + longueur minimale de barre-mere.
C   30-11-92  |DOWGHT2 | Support de la modelisation 1.2 du P.L.
C             |       | ( grand nombre de longueurs )
C   02-12-92  |DOWGHT  | Arrondi sur pourcent fixe a ajouter et poids.
C   09-12-92  |DOWGHT3 | Support de la modelisation 1.3 du P.L.
C             |       | ( combinaisons par fenetre sur les longueurs )
C   17-12-92  |DOWGHT4 | Orchestration des modules inherents a la CAEB.
C  [change_entry]
```

```
C ================
C  CONSTANTES
C ================

      IMPLICIT NONE
      INCLUDE 'ORCHESTR.BLK'


C ================
C  VARIABLES
C ================

C     POIDS0:  UN CANDIDAT POIDS LINGOT.
C     I, J, J1, K:  INDICES DES COMBINAISONS, POIDS LINGOTS (2) ET LONGUEURS
C              DE POUTRELLES RESPECTIVEMENT.
C     N:       INDICE DE BOUCLE.
C     ECART:   ECART MINIMUM VOULU ENTRE DEUX POIDS LINGOTS CANDIDATS


      INTEGER*2  I, J, J1, K, N, DICHO

      REAL    POIDS0, ECART



C ================
C  MAIN
C ================


C     *VERIFICATION DES LONGUEURS DES COMB*

1070  FORMAT ('-',9X,<WIN_SIZE>(I2.1,'xL',I2.2,' '))
      DO 100 I=1, NB_COMB
     POIDS0=LG_COMB(I)
     IF (POIDS0.GT.LMAX_BM .OR. POIDS0.LT.LMIN_BM) THEN
      WRITE(*,*)
      WRITE(*,*) '     non valide:',POIDS0,' metres.'
     ENDIF
C  WRITE(*,1070) ((COMB(I,K), K+SHIFT(I)+PART_FRST_LG-1),
C     >                          K=1,WIN_SIZE)
C  IF (POIDS0.GT.LMAX_BM) THEN
C    WRITE(*,*)
C  ENDIF
100   CONTINUE
      WRITE(*,*) ' (',NB_COMB,' combinaisons)'



C     *CALCUL DES POIDS LINGOTS CANDIDATS*
      NB_POIDS = 0
      DO 110 I=1, NB_COMB
     POIDS0 = LG_COMB(I)
C       *AJOUTER POIDS0 ET POIDS0 AUGMENTE SI NON DEJA EXISTANTS*
     DO 130 N=1,2
C          *AJOUTER POIDS0 SI NON DEJA EXISTANT*
       POIDS0 = INT(POIDS0*1000+1.0) / 1000.0
```

```
      J=DICHO(POIDS0,POIDS,NB_POIDS)
      IF (J.GT.0 .AND. POIDS0.LE.LMAX_BM .AND. POIDS0.GE.LMIN_BM)
     >       THEN
        DO 140 J1=NB_POIDS,J,-1
             POIDS(J1+1)=POIDS(J1)
140          CONTINUE
        POIDS(J)=POIDS0
        NB_POIDS=NB_POIDS+1
      ENDIF
C         *AJOUTER POIDS0 AUGMENTE SI NON DEJA EXISTANT*
      POIDS0=(1+POURCENT)*POIDS0
130     CONTINUE
110   CONTINUE


C     *REGROUPEMENT DES LINGOTS SEMBLABLES*
      WRITE(*,*)
      WRITE(*,*) ' Nombre de poids lingots candidats: ', NB_POIDS
      WRITE(*,*) ' Ecart minimum (en metres) voulu entre 2 poids lingot
     >s ? '
      READ(*,*) ECART
      CALL GROUP(POIDS,NB_POIDS,ECART)


      RETURN
99999 END



C ================
C   DICHO
C ================


      FUNCTION DICHO(POIDS0,POIDS,N_POIDS)

C     ABSTRACT:
C           Retourne l'indice du tableau POIDS(1:N_POIDS) correspondant a
C           l'emplacement ou doit se placer la valeur POIDS0.
C           Retourne 0 si POIDS0 se trouve deja dans le tableau POIDS.
C           Le tableau POIDS doit etre trie dans l'ordre DECROISSANT.

      IMPLICIT NONE

C     DICHO:    INDICE DE POIDS0 DANS LE TABLEAU POIDS (0 SI INEXISTANT)
C     N_POIDS:  NBRE DE POIDS LINGOTS CANDIDATS.
C     POIDS:    CANDIDATS POIDS LINGOT.
C     POIDS0:   UN CANDIDAT POIDS LINGOT.
C     FOUND:    BOOLEAN D'EXISTANCE DE POIDS0 DANS POIDS.
C     G,D,M:    INDICES DE DICHOTOMIE.

      INTEGER*2 DICHO, N_POIDS, G, D, M
      REAL      POIDS(*), POIDS0
      LOGICAL   FOUND

      FOUND=.FALSE.
      G=1
```

```
      D=N_POIDS
150   IF (D.GE.G .AND. .NOT.FOUND) THEN
      M=(D+G)/2
      IF (POIDS(M) - POIDS0) 10,20,30
10      D=M-1
      GOTO 40
20      FOUND=.TRUE.
      GOTO 40
30      G=M+1
40      GOTO 150
      ENDIF
      IF (.NOT.FOUND) THEN
      DICHO=G
      ELSE
      DICHO=0
      ENDIF

      RETURN
      END



C ================
C   GROUP
C ================


      SUBROUTINE GROUP(POIDS,N_POIDS,ECART)

C     ABSTRACT:
C          Le tableau POIDS(1:N_POIDS) suppose TRIE est renvoye tel que
C          deux de ses poids soient au minimum differents de ECART.
C          POIDS(1) est toujours conserve.
C          Le tableau renvoye est encore trie dans le meme sens.
C          N_POIDS est mis a jour.

      IMPLICIT NONE

C     N_POIDS:  NBRE DE POIDS LINGOTS CANDIDATS.
C     POIDS:    CANDIDATS POIDS LINGOT.
C     X, Y, J:  INDICES.
C     ECART:    ECART MINIMUM VOULU ENTRE DEUX POIDS

      INTEGER*2 N_POIDS, X, Y, J
      REAL      POIDS(*), ECART

      X=1
      Y=2
160   IF (Y.LE.N_POIDS) THEN
      IF (ABS(POIDS(X)-POIDS(Y)).GT.ECART) THEN
C          *CONSERVER POIDS(Y)*
       X=Y
       Y=Y+1
      ELSE
C          *SUPPRIMER POIDS(Y)*
       DO 170 J=Y,N_POIDS
            POIDS(J)=POIDS(J+1)
170        CONTINUE
```

```
      N_POIDS=N_POIDS-1
    ENDIF
    GOTO 160
      ENDIF

      RETURN
      END
```

Le module SOLUTION.FOR

```
      SUBROUTINE Solution

C
C ABSTRACT:
C
C       Extraire la solution du Programme Lineaire de CAEB de son fichier
C       de rapport.
C
C INPUTS:
C
C       Fichier CAEB.XPR genere par le P.L.
C
C OUTPUTS:
C
C       CPLE_NCOMB, CPLE_LCOMB, CPLE_NBCOMB,
C       CPLE_WGHT, CPLE_NWGHT, CPLE_NBWGHT
C
C BODY:
C
C
C AUTHOR(S):
C
C       J-P Castiaux
C
C CREATION DATE:
C
C       23-11-92
C
C MODIFICATION HISTORY:
C
C       Date    | Name  | Description
C ---------------+-------+-----------------------------------------------
C    23-11-92   |SOLUTION| Programme complet
C    01-12-92   |SOLUTN2 | Support de la modelisation 1.2 - grd nbre de lg.
C    10-12-92   |SOLUTN3 | Support de la modelisation 1.3 - fenetre sur lg.
C    23-12-92   |SOLUTN4 | Orchestration des differents modules.
C    05-01-93   |SOLUTN5 | Sous-routine U_SOLUTN ajoutee.
C  [change_entry]
```

```
C ================
C  VARIABLES
C ================

C    NBCB:    NOMBRE DE FOIS QUE LES COMB SONT UTILISEES.
C    NBCB0:   NOMBRE DE FOIS QU'UNE COMB EST UTILISEE.
C    POIDS0:  UN CANDIDAT POIDS LINGOT.
C    NBPDS:   NOMBRE DE FOIS QUE LES POIDS SONT UTILISES.
C    NBPDS0:  NOMBRE DE FOIS QU'UN POIDS EST UTILISE.
C    I, J, K: INDICES DES COMBINAISONS, POIDS LINGOTS ET LONGUEURS
C             DE POUTRELLES RESPECTIVEMENT.
C    N:       INDICE DE BOUCLE.
C    LINE:    UNE LIGNE DU FICHIER CAEB.XPR.
C    STOCK:   LONGUEUR TOTALE DES POUTRES PRODUITES EN SURPLUS.
C    MITR:    LONGUEUR EFFECTIVE - LONGUEUR POUTRELLES PRODUITES.
C    OBJ:     LONGUEUR EFFECTIVE.
C    NLONG:   NOMBRE DE POUTRELLES A PRODUIRE PAR LONGUEUR.


     IMPLICIT NONE

     INCLUDE 'ORCHESTR.BLK'

     INTEGER*2 NBCB(PL_MAX_COMB), NBPDS(PL_MAX_COMB),
    >          NLONG(PL_MAX_NLONG), I, J, K, N

     REAL      POIDS0, NBCB0, NBPDS0, STOCK, MITR, OBJ

     CHARACTER*75 LINE


C ================
C  MAIN
C ================


     WRITE(*,*)
     WRITE(*,*) 'RAPPORT PARTITION COURANTE:'
     WRITE(*,*) '============================'
     WRITE(*,*)
1040 FORMAT (1X,A26,I4.1)
     WRITE(*,1040) ' Nbre de longueurs:      ', PART_NLONG
     WRITE(*,*)
     WRITE(*,1040) ' Nbre de combinaisons:   ',NB_COMB
     WRITE(*,*)
     WRITE(*,*) ' Nbre de poids lingots candidats:',NB_POIDS


C    *LIRE LES QUANTITES DE POUTRELLES*
     OPEN(10,FILE='NLONG.XPR',STATUS='OLD')
     READ(10,*) (NLONG(K), K=1,PART_NLONG)
     CLOSE(10)


C    *LIRE LA VALEUR DE LA FONCTION OBJECTIVE*
```

```
      OPEN(10,FILE='CAEB.XPR',STATUS='OLD')
1050  FORMAT(A75)
110   READ(10,1050) LINE
      LINE=LINE(1:16)
      IF (LINE.NE.'Optimal solution') THEN
     GOTO 110
      ENDIF
1000  FORMAT(A28,F20.5)
      READ(10,1000) LINE, OBJ
      IF (LINE.NE.'Objective function value is ') THEN
     WRITE(*,*) 'Valeur de la Fonction Objective non trouvee!'
      ENDIF


C     *LIRE LES NCOMB*

      WRITE(*,*)
      WRITE(*,*) ' Combinaisons de decoupe:'
      WRITE(*,*)
115   READ(10,1050) LINE
      IF (LINE(1:18).NE.'  Number   Column') THEN
     GOTO 115
      ENDIF

2000  FORMAT(11X,A8,5X,F14.2)
2005  FORMAT('-',4X,I3.1,' x ',A7,' : ')
2010  FORMAT('-',' ',I2.1,'xL',I2.2)
      DO 120,I=1,NB_COMB
     READ(10,2000) LINE, NBCB0
     NBCB(I)=INT(NBCB0)
     IF (NBCB(I).NE.0) THEN
       CPLE_NCOMB = CPLE_NCOMB+1
       CPLE_SHIFT(CPLE_NCOMB) = SHIFT(I) + PART_FRST_LG -1
       CPLE_NBCOMB(CPLE_NCOMB) = NBCB(I)
       CPLE_LGCOMB(CPLE_NCOMB) = LG_COMB(I)
       WRITE(*,2005) NBCB(I),LINE(2:8)
       DO 117,K=1,WIN_SIZE
        CPLE_COMB(CPLE_NCOMB,K) = COMB(I,K)
        IF (COMB(I,K).NE.0) THEN
              WRITE(*,2010) COMB(I,K), K+SHIFT(I)+PART_FRST_LG-1
        ENDIF
117         CONTINUE
     ENDIF
120   CONTINUE


C     *LIRE LES NPOIDS*

      WRITE(*,*)
      WRITE(*,*) ' Commande de lingots:'
      WRITE(*,*)
      DO 130,J=1,NB_POIDS
     READ(10,2000) LINE, NBPDS0
     NBPDS(J)=INT(NBPDS0)
130   CONTINUE

3000  FORMAT (10X,I4.1,' x POIDS',A3,' : ',F8.5,' metres')
```

```
      DO 140,J=1,NB_POIDS
     READ(10,2000) LINE, NBPDS0
     NBPDS(J)=NBPDS(J)+INT(NBPDS0)
     IF (NBPDS(J).NE.0) THEN
       WRITE(*,3000) NBPDS(J),LINE(6:8),POIDS(J)
       CPLE_NWGHT=CPLE_NWGHT+1
       CPLE_NBWGHT(CPLE_NWGHT)=NBPDS(J)
       CPLE_WGHT(CPLE_NWGHT)=POIDS(J)
     ENDIF
140   CONTINUE

      CLOSE(10)


C     *STATISTIQUES*

C      *Nbre de poutrelles*
      WRITE(*,*)
      WRITE(*,*) ' Nombre de poutrelles produites:'
      WRITE(*,*)
      STOCK=0.0
4000  FORMAT (10X,I4.1,' x ',F8.5,' metres  (+',I4.1,')')
      DO 150, K=1,PART_NLONG
     N=0
     DO 160,I=1,NB_COMB
       IF (K.GT.SHIFT(I) .AND. K.LE.SHIFT(I)+WIN_SIZE) THEN
            N=N+NBCB(I)*COMB(I,K-SHIFT(I))
       ENDIF
160      CONTINUE
     STOCK = STOCK + (N-NLONG(K)) * CPLE_LONG(K+PART_FRST_LG-1)
     WRITE(*,4000) N,CPLE_LONG(K+PART_FRST_LG-1),N-NLONG(K)
     CPLE_QOK(K+PART_FRST_LG-1)=CPLE_QOK(K+PART_FRST_LG-1)+N
150   CONTINUE
5000  FORMAT (35X,'(+',F10.3,' metres)')
      WRITE(*,5000) STOCK

C      *Acier*
      WRITE(*,*)
6000  FORMAT (1X,' Longueur d''acier effective:',F10.3,' metres.')
      WRITE(*,6000) OBJ
      OBJ=0
      DO 170,J=1,NB_POIDS
     OBJ=OBJ+NBPDS(J)*POIDS(J)
170   CONTINUE
7000  FORMAT (1X,' Verification:',F10.3,' metres.')
      WRITE(*,7000) OBJ
      OBJECTIF=OBJECTIF+OBJ

C      *Mitraille*
      MITR=OBJ
      DO 180, K=PART_FRST_LG,PART_LST_LG
     MITR=MITR-NLONG(K-PART_FRST_LG+1)*CPLE_LONG(K)
180   CONTINUE
      MITR=MITR-STOCK
8000  FORMAT (1X,' Mitraille (lg. eff. - prod.): ',F10.3,' metres.')
      WRITE(*,*)
      WRITE(*,8000) MITR
```

```
C     *BM*
      N=0
      DO 190,J=1,NB_POIDS
     N=N+NBPDS(J)
190   CONTINUE
      WRITE(*,1040) ' Nombre de barres-meres:  ',N

99999 END


      SUBROUTINE U_Solutn


C
C ABSTRACT:
C
C       Extraire la solution du Programme Lineaire d'UNIFORMisation des
C     - poids lingots.
C
C INPUTS:
C
C       Fichier CAEB.XPR genere par le P.L. UNIFORM
C
C OUTPUTS:
C
C       CPLE_WGHT, CPLE_NWGHT, CPLE_NBWGHT
C
C BODY:
C
C
C AUTHOR(S):
C
C       J-P Castiaux
C
C CREATION DATE:
C
C       05-1-93
C
C MODIFICATION HISTORY:
C
C       Date     | Name  | Description
C ----------------+-------+-----------------------------------------------
C  [change_entry]



C ================
C  VARIABLES
C ================

C     NBCB:     NOMBRE DE FOIS QUE LES COMB SONT UTILISEES.
C     NBCB0:    NOMBRE DE FOIS QU'UNE COMB EST UTILISEE.
C     POIDS0:   UN CANDIDAT POIDS LINGOT.
C     NBPDS:    NOMBRE DE FOIS QUE LES POIDS SONT UTILISES.
C     NBPDS0:   NOMBRE DE FOIS QU'UN POIDS EST UTILISE.
C     I, J, K:  INDICES DES COMBINAISONS, POIDS LINGOTS ET LONGUEURS
C               DE POUTRELLES RESPECTIVEMENT.
C     N:        INDICE DE BOUCLE.
```

```
C     LINE:    UNE LIGNE DU FICHIER CAEB.XPR.
C     STOCK:   LONGUEUR TOTALE DES POUTRES PRODUITES EN SURPLUS.
C     MITR:    LONGUEUR EFFECTIVE - LONGUEUR POUTRELLES PRODUITES.
C     OBJ:     LONGUEUR EFFECTIVE.
C     NLONG:   NOMBRE DE POUTRELLES A PRODUIRE PAR LONGUEUR.


      IMPLICIT NONE

      INCLUDE 'ORCHESTR.BLK'

      INTEGER*2 NBCB(PL_MAX_COMB), NBPDS(PL_MAX_COMB),
     >          NLONG(PL_MAX_NLONG), I, J, K, N

      REAL      POIDS0, NBCB0, NBPDS0, STOCK, MITR, OBJ

      CHARACTER*75 LINE



C ==================
C  MAIN
C ==================


      WRITE(*,*)
      WRITE(*,*) '==================================='
      WRITE(*,*) 'UNIFORMISATION DES POIDS LINGOTS:'
      WRITE(*,*) '==================================='
      WRITE(*,*)
1040  FORMAT (1X,A36,I4.1)
      WRITE(*,1040) ' Nbre de poids lingots candidats:',NB_POIDS


C     *LIRE LA VALEUR DE LA FONCTION OBJECTIVE*

      OPEN(10,FILE='CAEB.XPR',STATUS='OLD')
1050  FORMAT(A75)
110   READ(10,1050) LINE
      LINE=LINE(1:16)
      IF (LINE.NE.'Optimal solution') THEN
     GOTO 110
      ENDIF
1000  FORMAT(A28,F20.5)
      READ(10,1000) LINE, OBJ
      IF (LINE.NE.'Objective function value is ') THEN
     WRITE(*,*) 'Valeur de la Fonction Objective non trouvee!'
      ENDIF
115   READ(10,1050) LINE
      IF (LINE(1:18).NE.'  Number   Column') THEN
     GOTO 115
      ENDIF


C     *LIRE LES NPOIDS*

      WRITE(*,*)
      WRITE(*,*) ' Commande de lingots:'
```

```
      WRITE(*,*)
2000  FORMAT(11X,A8,5X,F14.2)
      DO 130,J=1,NB_POIDS
     READ(10,2000) LINE, NBPDS0
     NBPDS(J)=INT(NBPDS0)
130   CONTINUE

3000  FORMAT (10X,I4.1,' x POIDS',A3,' : ',F8.5,' metres')
      DO 140,J=1,NB_POIDS
     READ(10,2000) LINE, NBPDS0
     NBPDS(J)=NBPDS(J)+INT(NBPDS0)
     IF (NBPDS(J).NE.0) THEN
       WRITE(*,3000) NBPDS(J),LINE(6:8),POIDS(J)
       CPLE_NWGHT=CPLE_NWGHT+1
       CPLE_NBWGHT(CPLE_NWGHT)=NBPDS(J)
       CPLE_WGHT(CPLE_NWGHT)=POIDS(J)
     ENDIF
140   CONTINUE

      CLOSE(10)


C     *STATISTIQUES*

C      *Acier*
      WRITE(*,*)
6000  FORMAT (1X,' Longueur d''acier effective:',F10.3,' metres.')
      WRITE(*,6000) OBJ
      OBJ=0
      DO 170,J=1,NB_POIDS
     OBJ=OBJ+NBPDS(J)*POIDS(J)
170   CONTINUE
7000  FORMAT (1X,' Verification:',F10.3,' metres.')
      WRITE(*,7000) OBJ
      OBJECTIF=OBJECTIF+OBJ

C      *Mitraille*
      MITR=OBJ
      DO 180, K=PART_FRST_LG,PART_LST_LG
     MITR=MITR-NLONG(K-PART_FRST_LG+1)*CPLE_LONG(K)
180   CONTINUE
      MITR=MITR-STOCK
8000  FORMAT (1X,' Mitraille (lg. eff. - prod.): ',F10.3,' metres.')
      WRITE(*,*)
      WRITE(*,8000) MITR

C      *BM*
      N=0
      DO 190,J=1,NB_POIDS
     N=N+NBPDS(J)
190   CONTINUE
      WRITE(*,1040) ' Nombre de barres-meres:  ',N

99999 END
```

Le module UNIFORM.FOR

```
      SUBROUTINE Uniform

C
C ABSTRACT:
C
C       Uniformisation par un Programme Lineaire UNIFORM des poids lingots
C       qui sont la solution par le P.L. CA_xx des partitions du couple
C       traite dans l'application CAEB.
C
C INPUTS:
C
C       CPLE_NCOMB, CPLE_LCOMB
C
C OUTPUTS:
C
C       NB_POIDS, POIDS
C
C BODY:
C
C
C AUTHOR(S):
C
C       J-P Castiaux
C
C CREATION DATE:
C
C       05-01-93
C
C MODIFICATION HISTORY:
C
C       Date    | Name  | Description
C ---------------+-------+-----------------------------------------------------
C               |       |
C  [change_entry]



C ==================
C    GLOBAL
C ==================

      IMPLICIT NONE
      INCLUDE 'ORCHESTR.BLK'


C ==================
C  VARIABLES
C ==================

C     I1, K:      Variable integer de travail.

      INTEGER*2 I1, K, TO_BE_DELETED
```

```
C ================
C  MAIN
C ================


C     * PREPARATION DES POIDS LINGOTS CANDIDATS*

  NB_COMB = CPLE_NCOMB
  DO 1000, K=1,NB_COMB
   LG_COMB(K) = CPLE_LGCOMB(K)
1000   CONTINUE
  CALL DOWEIGHT

C     * CREATION DES FICHIERS EXPRESS *
C      *dim.xpr*
  OPEN(10,FILE='U_DIM.XPR',STATUS='UNKNOWN')
230    FORMAT (I3.3,',',I3.3,',',I2.2)
  WRITE(10,230) NB_COMB, NB_POIDS, MAX_PDS
  CLOSE(10)

C      *u_lcomb.xpr*
  OPEN(10,FILE='U_LCOMB.XPR',STATUS='UNKNOWN')
233    FORMAT(1X,F7.3)
  WRITE(10,233) (CPLE_LGCOMB(K), K=1,CPLE_NCOMB)
  CLOSE(10)

C      *u_ncomb.xpr*
  OPEN(10,FILE='U_NCOMB.XPR',STATUS='UNKNOWN')
235    FORMAT(1X,I3.3)
  WRITE(10,235) (CPLE_NBCOMB(K), K=1,CPLE_NCOMB)
  CLOSE(10)

C      *u_poids.xpr*
160    FORMAT (1X,F8.4)
C170   FORMAT ('-',<NB_POIDS>(8X,F8.4))
  OPEN(10,FILE='U_POIDS.XPR',STATUS='UNKNOWN')
  WRITE(10,160) (POIDS(I1), I1=1,NB_POIDS)
  CLOSE(10)
  WRITE(*,*) ' Nombre de poids lingots candidats:',NB_POIDS
C      WRITE(*,170) (POIDS(I1), I1=1,NB_POIDS)


C     * P.L. UNIFORMISATION *

      WRITE(*,*) '--------- TAPER "0 <ENTER>" APRES
     >TERMINAISON DU P.L. D"UNIFORMISATION---'
      READ(*,*) TO_BE_DELETED
      WRITE (*,*) 'OK.'

      RETURN
      END
```